Starting an Independent Software Business

MacTech Magazine
June • 2009

# MACTECH®

The Journal of Macintosh Technology

# DECIPHERING PKI

## Your Key to Certified Security

New Tools for Collaboration: Sharepoint 101

Handling User Preferences

Scripting Opportunities for System Administrators

MACTECH.COM

$8.95 US, $12.95 Canada

0 32128 74887 8 06

ISSN 1067-8360 Printed in U.S.A.

TOO MANY TO DOS?

# Record. Edit. Play.


Using WireTap Studio, you can record the discrete audio output of any application, as well as all system audio, or record audio input from any microphone, line-in, or audio input hardware.

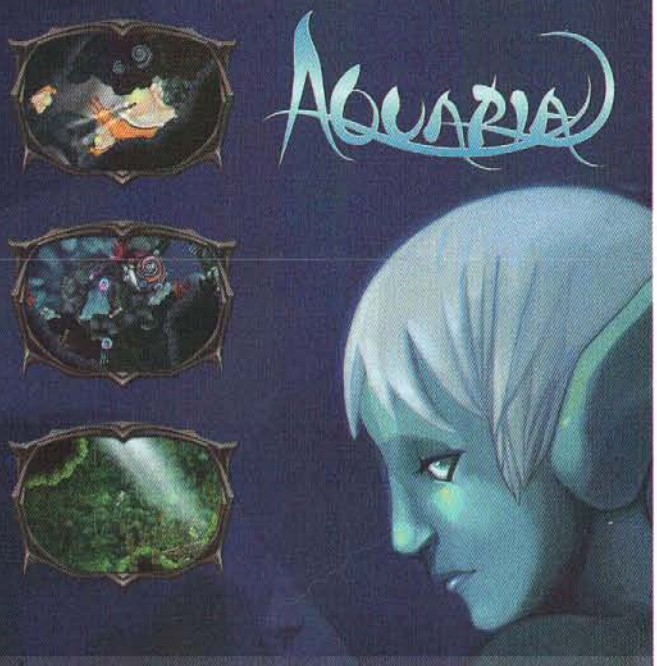
If you can hear it, WireTap Studio can record it.

## WireTap Studio

AMBROSIA SOFTWARE INC

Download your free trial now: http://www.AmbrosiaSW.com/mactech

# An underwater adventure awaits...




BIT BLOT

AMBROSIA SOFTWARE INC

INDEPENDENT GAMES FESTIVAL 2007 WINNER

Download your free trial now: http://www.AmbrosiaSW.com/mactech

# multiwinia

darwin is dead. prepare for war.

Balance bloodlust with strategy in this furiously-paced real time war game where you pit your armies of stick men against online opponents or single player to control the fractalized world of multiwinia.

AMBROSIA SOFTWARE INC

Download your free trial now: http://www.AmbrosiaSW.com/mactech

# Live sound effects on tap!

## soundboard

Create a library of your favorite audio clips that you can trigger with a tap.

Use Soundboard to enhance your podcasts or broadcasts with sound clips, effects, or musical accompaniment.

AMBROSIA SOFTWARE INC

Download your free trial now: http://www.AmbrosiaSW.com/mactech

# SnapzProX

Free Trial Version!

## If a picture is worth a thousand words, imagine how priceless a movie would be...

Snapz Pro X allows you to effortlessly record anything on your screen, saving it as a QuickTime® movie that can be emailed, put up on the web, or distributed however you want.


Why take a static screenshot when Snapz Pro X makes creating a movie just as easy?
In addition to a video capture engine that is 20 times faster than anything else on the market, Snapz Pro X has so many other new features, you'll quickly wonder how you ever lived without it!

**Download your free trial now:**
**http://www.AmbrosiaSW.com/MacTech**

AMBROSIA® SOFTWARE INC

2004 winner macworld eddys

Macworld

Mac

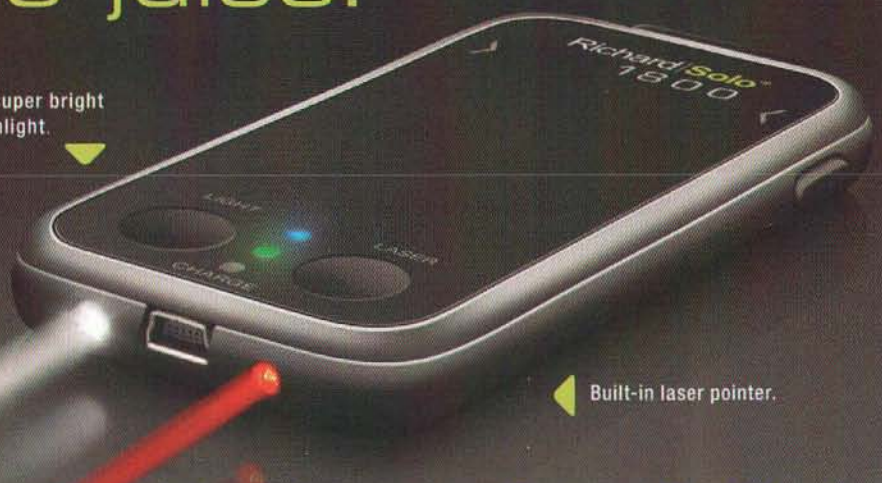
Universal

# Give your iPhone or BlackBerry® more than double the juice.

**For iPhone/iPod**
Direct plug-in/no cable.
Includes 2G/3G support brace.
RS001 **$69.95**

**For iPhone/iPod with Cable**
Cable connection only.
Perfect for iPod touch.
RS008 **$69.95**

**For All BlackBerry®**
Cable or direct plug-in.
Works with all USB-port phones.
RS007 **$69.95**




## The quick pocket solution to top up your iPhone or BlackBerry! And now get a free iPhone case too!

*by Richard Thalheimer*

Just plug RichardSolo 1800 into your iPhone/BlackBerry once or twice a day for fifteen minutes, and keep your device charged up. At your desk or at dinner, plug in RichardSolo to instantly transfer charge. No more battery worries! 1800 mAh lithium-ion battery is largest in its class and holds its charge for months. RichardSolo 1800 will charge iPhone to full 1.5 times, and lithium-ion is good for 3-5 years of recharges.

Retractable USB cable, 110-240v AC wall charger, and dual-port USB car charger included — like getting two extra chargers for free! Even charge the RichardSolo 1800 and phone together at the same time. Take only one charger when traveling and wake up in the morning with the RichardSolo and your phone charged. Put the RichardSolo in your pocket, plug it into your phone now and then to top it up. Use your phone while charging it.

Enjoy a built-in laser pointer and ultra-bright LED flashlight, too! These two lights use very little power, so you get handy extra features. RS001 is the only snap-on battery available with a latching mechanism for iPhone, to keep battery firmly attached. RS001 fits all iPhone/iPod cases except the very thickest armor. Optional 2G/3G support brace included.

**RS001** works with iPhone and all iPod models except shuffle.
**RS008** works with iPhone and all iPod models except shuffle — perfect for iPod touch.
**RS007** works with all BlackBerry models and any phone with a mini/micro USB charging port. Plug in directly or use with cable.

Your satisfaction is guaranteed, with our 30-day, no-hassle return privilege. If you're not satisfied for any reason, we'll email you a pre-paid return label. You have no risk whatsoever, and the warranty is now one full year.

**On-line ordering and blog reviews.**
***Order two of the same and save 15%***

Buy Me An iPhone.com BEST '08 iPhone Backup Battery

### We are here to support you! Actual customer comments:

"To have your company exhibit such exceptional service is unbelievably refreshing." — ***P.S.***

"Dear Richard and Team: This is what I call great customer support. I wish more companies would figure this out these days. Thank you so much." — ***D.C.***

"You have provided me one of the best services I have ever encountered on any on-line/telephone shopping." — ***T.K.***

### Reviewers give it top marks.

"Your product is excellent, and the customer service is, of course, outstanding." — ***Chris.Pirillo.com***

"Now RichardSolo has released a much improved version of the Backup Battery that not only provides more power for your iPhone or iPod, but some excellent new features that trash the competition." — ***TUAW.com***

"The quintessential accessory for all iPhone/ iPod owners. If you don't already own one, you need to order yours today." — ***BuyMeAniPhone.com***

Made for iPod  Works with iPhone



### Included Free:

- *AC Wall Charger*
- *Dual-port USB Car Charger*
- *Retractable USB Cable*
- *Support brace for iPhone*
- *Cable shown with model*

### Plus

Order now, and we will include our **$24.95** hard case for iPhone 3G **as a free bonus with RS001/RS008!**

Good protection, textured finish, works perfectly with our support brace.

**Order today, goes out next business day | www.RichardSolo.com | email: support@RichardSolo.com**

Richard|Solo®

# TABLE OF CONTENTS

# From the Editor

We definitely are living in interesting times. And difficult times. Thanks to global economic fluctuations, businesses large and small have been generally hurting. Consultants, which business can outsource to, tend to be doing well, and others are working for larger companies that are holding on. We encounter, and write to, people in each of these categories. And MacTech wants to help you excel no matter what you're doing.

Seems like people are letting us: year-to-date, MacTech is up 12% compared to the same period last year (2008). There are more articles, more ad pages and more readers. In fact, the issue you're reading now is the largest issue of MacTech since 2001, and it's larger than any other Mac-oriented Magazine in June.

When the competition is fierce, you can't have enough information. Many of you may be reading MacTech for the first time—or the first in a long time—thanks to WWDC. If you're at WWDC, you're doing so to better and advance yourself. That's what we look to do year round. Like WWDC, we cover many segments: IT/Consultants, Developers for OS X and iPhone and those new hybrids—Sys Admins that can develop.

Speaking of WWDC, make sure you get the most out of it. Of course, that means sitting in on sessions and speaking with Apple Engineers. Also, be sure to connect with your peers! This is possibly the most important part of the week. After the week is over, it's unlikely that you'll stay in close contact with any Apple employee that you meet. Nor can they give you personal support the remainder of the year. Hopefully, though, you'll continue to stay in touch with the peers you meet at the conference – be it through public mailing lists, regional meetings (like CocoaHeads, Apple Consultants Network members or other Macintosh User Group-type gatherings) or even directly.

So, what is on our collective mind this month? Well, a little bit of everything. As you can see by the cover, Public Key Encryption (PKI) is weighing on us. In the first of a series, Michele (Mike) Hjörleifsson gives us a gentle introduction to PKI for all of those that need to learn about this important topic.

New author Sengan Baring-Gould shows us a very interesting way to generate a backtrace of the stack for Obj-C. When your program dies a horrible death, and you need to figure out how you got to the point where it actually crashes, this is a great resource.

Michael Göbel and Oliver Pospisil continue their "Inspired by Life" column on starting an independent software company. Great ideas to be found, so, follow along!

This month's Mac in the Shell continues its "Learning Python on the Mac" series and delves into PyObjC, or, accessing native Cocoa and Obj-C using just Python. This column looks at accessing groups via the Address Book.

William Smith returns with another article describing ways to integrate OS X into a Microsoft shop. This time, he teaches all about the commonly found Sharepoint.

Back with another article in his series on software packaging is José Cruz. This month, he tackles a unique way of customizing the installer via a plug-in. Follow him through creating a plug-in using Xcode.

Greg Neagle, once again brings one for the Sys Admin playbook: methods and reasoning behind running scripts on end-user workstations. This is often an area that many system administrators struggle with. Never fear, Greg lays it out clearly.

In his Road to Code column, Dave Dribin covers saving user preferences using NSUserDefaults. Who doesn't like an application that remembers what they like?

There's more, but I'll wrap up by pointing out this month's MacTech Spotlight: Philip Goward and Greg Scown from SmileOnMyMac. SmileOnMyMac has been producing great utility software for OS X for a long time and has been through some ups and downs in the Mac market. We're happy to feature the co-founders of this company this month.

Enjoy the show; soak in the new knowledge. See you next month.

Edward Marczak,
Executive Editor

## Communicate With Us

### Department E-Mails

**Orders, Circulation, & Customer Service**
cust_service@mactech.com

**Press Releases**
press_releases@mactech.com

**Ad Sales**
adsales@mactech.com

**Editorial**
editorial@mactech.com
(Authors only, no pr)

**Accounting**
accounting@mactech.com

**Marketing**
marketing@mactech.com

**General**
info@mactech.com

**Web Site**
http://www.mactech.com

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail?**

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact MacTech Magazine Customer Service at 877-MACTECH

We love to hear from you! Please feel free to contact us with any suggestions or questions at any time.

Write to letters@mactech.com or editorial@mactech.com as appropriate.


# MACTECH®
The Journal of Macintosh Technology

A publication of XPLAIN CORPORATION





**Canada Post:** Publications Mail Agreement #41513541
**Canada Returns to be sent to:** Bleuchip International, P.O. Box 25542, London, ON N6C 6B2

**MacTech Magazine** (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 5776-D Lindero Canyon #189, Westlake Village, CA 91362. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are $47.00 per year. Canadian subscriptions are $59.00 per year. All other international subscriptions are $97.00 per year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

**POSTMASTER:** Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

by Edward Marczak

# Python on the Mac: PyObjC

## Writing native Cocoa apps using Python

## Introduction

Over the last few months, we've been covering the basics of Python. Aside from a few OS X-specific issues raised in the first article (how to get the built-in docs working, etc.), you could really take the lessons learned anywhere – Linux, Windows, or any platform where you find a Python runtime. We needed those basics – and we have more to cover, certainly. However, this is *MacTech*. There's plenty that one can do with some very basic Python and Python/Objective-C bridge, letting you tap into Cocoa. Cocoa? Isn't that reserved for Obj-C developers? Nope. While MacTech has covered this concept before (Scott Corely, "Python Cocoa: Delicious," February 2009), I'd like to put together the lessons learned in this column along with a more utilitarian approach.

## Read The Fine Manual

Anytime we're working with Cocoa and the technologies in OS X, we'll probably be pouring through the developer references at http://developer.apple.com. You'll need an ADC account to do so. Even the free variety will do, so, go sign up now if you haven't already!

Once you're logged into the Developer Connection, head to the developer docs at http://developer.apple.com/documentation/. More often than not, you'll search on the topic you're after. Sometimes, you find good documentation spread out over several categories. Today, we'll be looking at getting information out of Address Book. True to form, the docs are somewhat spread out. I'll make reference to each as I use it. In short, for now, just search on "address book".

## Translating Obj-C

First, why would we want to do this? There are certainly cases when developing for OS X where straight Obj-C is the right choice. However, I'm taking this from a System Administrator's point of view. Often, a System Administrator is already writing basic scripts in bash. I love bash, but there's only so far that it'll get you without becoming painful. If you're writing a script in bash and it passes the 4 functions milestone, it may be time to consider a language more suited to your task. For example, bash isn't really great with databases.

Sure, you can use the mysql binary, pipe the output to awk, and manipulate results from there. But is that the best use of your time and talent? Ever deal with arrays in bash? Pain. While I may recommend Python or Ruby as a step up in general, these languages are made even more special under OS X thanks to Apple's inclusion of an Obj-C bridge. BridgeSupport opens up OS X's native APIs to Python, Ruby and JavaScript. This is available and standard on every Mac running 10.5 or higher. (10.4 support is available, but you'll need to install it yourself, which is outside the scope of this article). BridgeSupport deals with all of the behind-the-scenes work of converting between Python and the native frameworks. The first challenge to this technique is interpreting the documentation. We're going to code all of this in Python, and the docs are directed at people writing in C and Objective-C. Anyone remember having to translate Mac Toolbox API calls from Pascal to C? I digress…

Now that we've covered Python classes, you know about sending a message to an object using dot notation. In last month's column, the BankClass example class contained deposit and withdraw methods. A new class could be created and a method called in the following manner:

```
acct = Account('Joan', 'Smith')     # Create new account
acct1.Deposit(50)        # Note use of class method here
```

However, if we got this information from Apple's developer documentation, you'd see something like this:

```
[acct1 Deposit:50]
```

This was covered in depth in the "Python Cocoa: Delicious" article referenced earlier, but I'll cover the basic rules here.

As you can see, Obj-C uses square brackets to send messages to objects. The easiest call to translate is a simple message with no parameters. This:

```
[object message];
```

in Python becomes:

```
object.message()
```


# Navigating the mobile world
## just got easier with iCooper.

The demands to capture, organize, and retrieve dynamic data in a mobile environment are rapidly increasing. One company, iCooper, gets it.

As the industry leader in dynamic data mobility, we develop and publish software that leverages the growing power of the iPhone, enabling enterprises and individuals to optimize their activities, workflow and productivity. Applying critical thinking and new ideas to the way users relate to data, iCooper creates simple, intelligent, mobile solutions to improve the lives of our customers.

ERP on a mobile platform? We can do that.
Mobile design and manufacturing integration? We can do that.
If data needs to move with you, move to iCooper.

Visit icooper.com for our story or call us at 360.844.5807 and tell us yours.

iCooper

When a method takes parameters, Obj-C places them in-line:

```
[object message:40 key:50];
```

Python keeps its usual format here, separating the method name and parameters. Each message and parameter gains a trailing underscore character:

```
object.message_key_(40,50)
```

Essentially, each colon is replaced by an underscore – even if there's only one parameter. For example:

```
object.message_(40)
```

To instantiate an objective-c class in the first place is fairly straightforward.

```
object = NSObject.alloc().init()
```

Let's see all of this in action.

## Reading the Address Book

The beauty of using a language like Python is that you can author in any editor you like, save and run. This skips the compile/link phase so familiar to Obj-C developers. So, pull up your favorite editor—remember, too, that most editors will be able to recognize Python code and syntax color, indent properly and so on, for you—and let's go.

Contained in /System/Library/Frameworks/Python.framework/ are the modules that Python uses for BridgeSupport. These can simply be imported into Python. First thing is first, our magic shebang line:

```
#!/usr/bin/env python
```

(Remember, if you have multiple versions of python on your system for some reason, under 10.5, the built-in BridgeSupport only works with Python 2.5. If you need you need to explicitly call that version, then do so). From here, we'll import the AddressBook framework:

```
from AddressBook import *
```

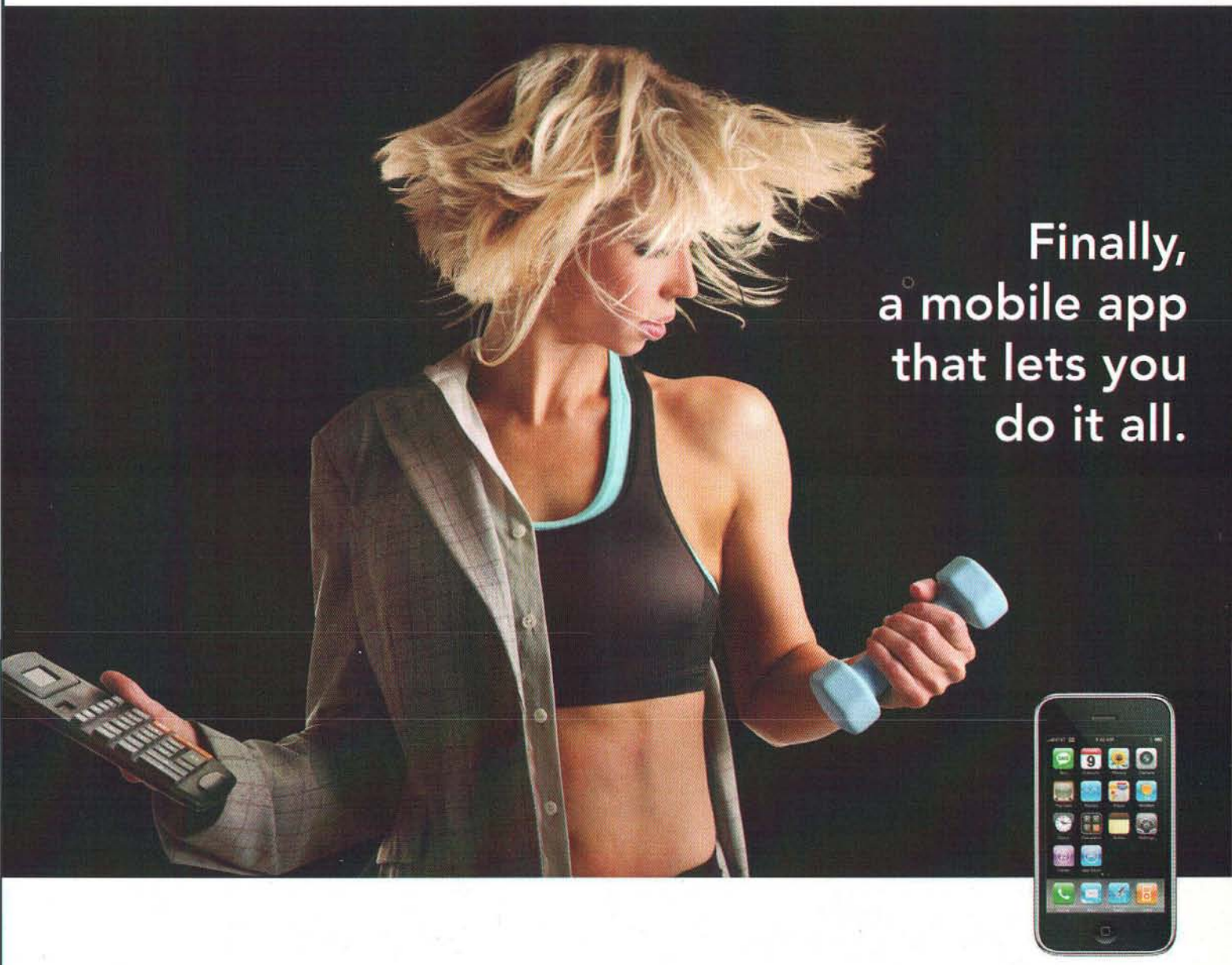
It's rare that I like or use the 'from blah import *' style, but there are times when it makes perfect sense. This, I feel, is one of them. We talked extensively about imports and namespaces in previous articles.

Let's create a new instance of an address book object:

```
aBook = ABAddressBook.sharedAddressBook()
```

Painless, right? This returns the address book for the logged-in user. Keeping this simple, let's grab the 'me' card for the logged in user and print it out:

```
myRecord = aBook.me()
print myRecord
```




The wait is almost over. A robust, feature packed app that lets you track, capture and organize data critical to your business, family or personal life.

The power of iPhone. The genius of iCooper. Workflow and productivity solutions for enterprises and individuals in the palm of your hand.

mobile warrior

See what all the fuss is about.
**mobilewarrior.com**

That's it! In 3 lines of code, we get a good amount of information. Here's the output:

```
ABPerson (0x1ab0a40) {
    ABPersonFlags  : 0
    ABRelatedNames : (
        "  child  Edward R Marczak
)
    Address        : (
        "  work  {
    City = Anytown;
    Country = USA;
    CountryCode = us;
    State = AA;
    Street = "555 Any Street";
    ZIP = 11111;
}
)
    AIMInstant     : (
        "  home  myaim
)
    Creation       : 2005-10-28 09:45:40 -0400
    Email          : (
        "  work  marczak@radiotope.com
)
    First          : Edward
    JobTitle       : Owner
    Last           : Marczak
    Middle         : R
    Modification   : 2009-01-14 11:11:25 -0500
    Organization   : Radiotope
    Phone          : (
        "  mobile  555-555-5185
        home    555 555-5370
        main    555-555-5489
)
    Title          : Mr.
    Unique ID      : B3AD0F6B-4AB8-4E84-82C4-
BF1EB7475659:ABPerson
}
```

Each of the properties in the record can be accessed and iterated over individually. Each property has a unique name used for this purpose. An illuminating method of discovering this, besides the Apple documentation is to use the `dir()` function that we've seen previously. Save your work and open a new document that contains this simple code:

```
#!/usr/bin/env python

import AddressBook

x = dir(AddressBook)
for i in x:
  print i
```

When you run it, you'll get an absolute ton of output, so pipe it through `less` or use a GUI editor that can run the code in its own window. It'll look like this:

```
ABACE
ABACL
ABAccessibilityMockUIElement
ABAddPropertiesAndTypes
ABAddRecord
ABAddToGroupCommand
ABAddressAttributedString
ABAddressBook
...
kABAIMHomeLabel
kABAIMInstantProperty
kABAIMWorkLabel
kABAddressCityKey
kABAddressCountryCodeKey
kABAddressCountryKey
kABAddressHomeLabel
```

```
kABAddressProperty
kABAddressStateKey
...
kCFXMLTreeErrorLocation
kCFXMLTreeErrorStatusCode
kEventABPeoplePickerDisplayedPropertyChanged
kEventABPeoplePickerGroupDoubleClicked
kEventABPeoplePickerGroupSelectionChanged
kEventABPeoplePickerNameDoubleClicked
kEventABPeoplePickerNameSelectionChanged
kEventABPeoplePickerValueSelectionChanged
kEventClassABPeoplePicker
kEventParamABPickerRef
objc
protocols
super
```

This lists every function and constant definition in the framework. In this case, we're interested in the block where each constant has the 'kAB' prefix. Each of these properties represents a potential field in the address book record – not all must be present. So, how can we tell which fields are present in a given record? We can ask. Back to our original code!

Here's a complete Python solution to dumping the current user's Address Book, I'll explain the parts not yet covered after this code listing.

***Listing 1: dumpAB.py***

```
#!/usr/bin/env python

from AddressBook import *

aBook = ABAddressBook.sharedAddressBook()
for person in aBook.people():
  properties = person.allProperties()
  for prop in properties:
    if prop == "com.apple.ABPersonMeProperty":
      continue
    elif prop == "com.apple.ABImageData":
      continue
    print prop, ":", person.valueForProperty_(prop)

  print '-'*60
  print
```

The `people()` method returns an array (an NSArray, specifically—the Obj-C Bridge deals with converting between the Obj-C types and Python types). We've previously covered Python for loops, and this one is no different. This loop iterates over each entry returned by the `people()` method, assigning it to `person` in each iteration.

With each person, we use the `allProperties()` method to determine the properties contained in that record. Then, we use another for loop to print only those properties. Note the `if` statement in this block: there are two properties present in each record that we're really not going to do anything with. Using a `continue` statement lets us restart the loop at the top.

Now, this isn't going to win any coding competitions, but look at how simple it is. No compiler or special IDE was needed to generate or run any of this.

## What Happened? (Maybe)

Some of you may have seen an error pop up while running this program. Something about a "UnicodeDecodeError". What happened? This, partially, is the old-school Unix ASCII-ness colliding with modern sensibilities. You'll only see this error if one of your address book entries has Unicode characters in it (accent marks, Asian/Hebrew/Russian character sets and so on). Well, OS X is built to deal with this. Now, this depends on the environment in which you ran this. Terminal.app should actually have no problem as it's Unicode compliant. Surprisingly, some GUI text editors still don't handle Unicode properly, or, just need a little help. One thing you can do is give the interpreter a little hint: immediately following the magic shebang line (`#!/usr/bin/env python`), include the following:

```
# encoding: utf-8
```

This explicitly sets the encoding of the document. Additionally, Python itself has built-in support for Unicode strings. When printing a string, prefix it with 'u' to specify Unicode output. Like this:

```
print u'This is a Unicode string'
```

If you're printing a variable, it's similarly easy:

```
print u'%s' % (variable)
```

This is just one of those things that OS X users expect, and script authors need to bear in mind. Kind of like spaces in filenames...

## Conclusion

There are actually a few more things we can cover about the Obj-C Bridge and its use in Python. However, we accomplished our goal for this month, and I hope you can see how easy some of these basic tasks are. You'll find that there are often several ways of approaching the code when using BridgeSupport. The methods used in this article are the most appropriate for the task at hand. See the References section below for the specific AddressBook documentation that I used to determine the bulk of this.

If we were more ambitious here, we could certainly do more with the data returned. Like write it out as a CSV file. AddressBook also supports group information, which I actually use fairy often, but that's a topic for next month.

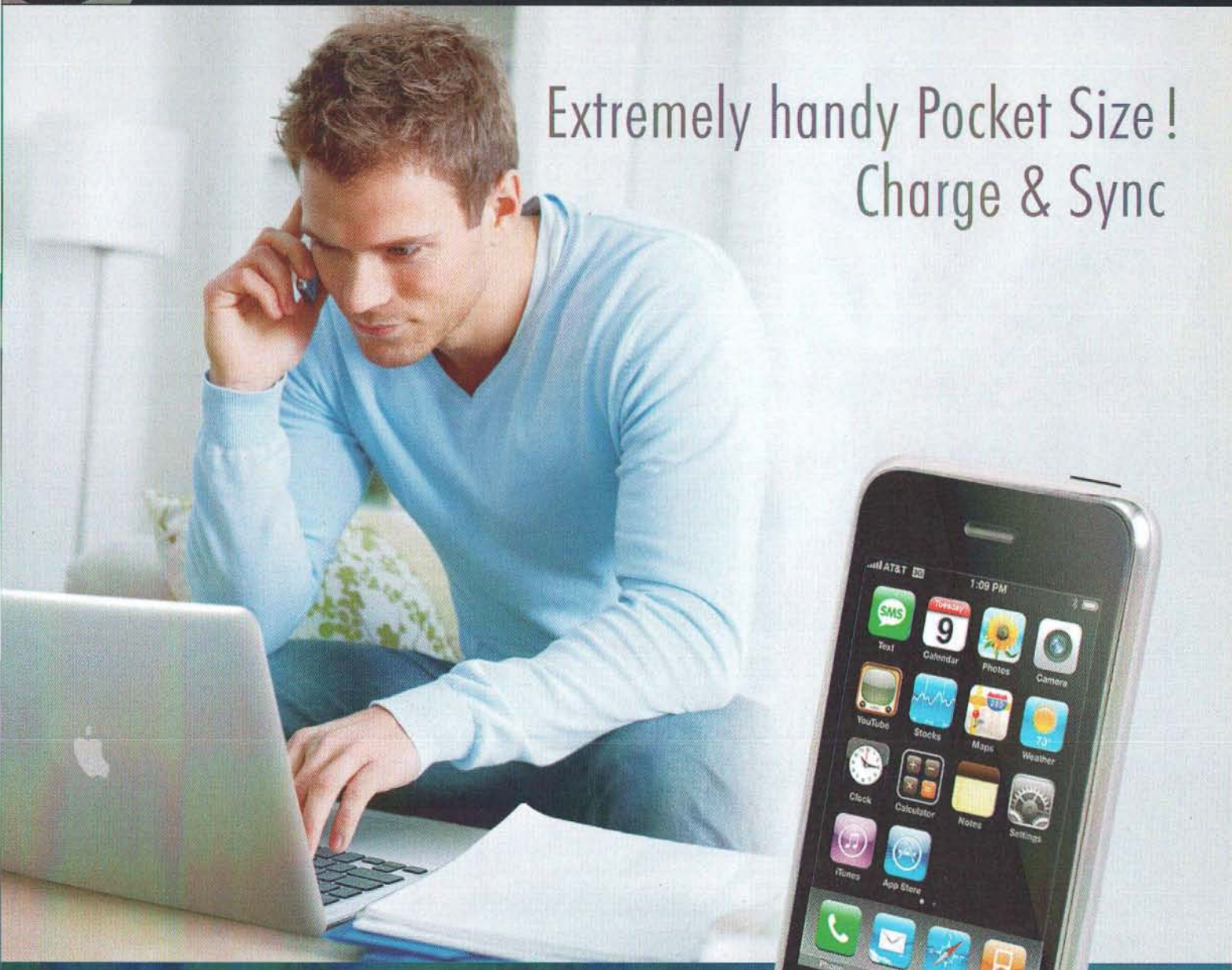
Media of the month: I know, I usually suggest a good book, movie or music CD here, but this month is a little different. This month's suggestion is the outdoors – don't forget about it! Seriously, I'm not really a 'sun person,' but it is nice to take a walk with no laptop/phone/electronic device. Take a bike ride. Have a picnic. Take a (real) hike. Experience it. Just don't forget that there's a world outside of the LCD that we often sit a foot or two away from.

Hopefully, you're reading this at Apple's (sold out, again!) WWDC. Most of us from MacTech are here too (and you may have received this issue while on line for the Keynote – welcome!). Ping us, stop us in the halls – just say hello! See you next month.

dexim™ *Put your dreams in life.*

www.dexim.net

# Extremely handy Pocket Size!
# Charge & Sync






## *Premium* MHub Dock Station

Designed for Mac/PC/iPhone/iPod/BlackBerry

- 3 Standard USB 2.0 ports
- Charge & Sync dock station for iPhone / iPod / BlackBerry
- Hi-speed HCSD / SD / MMC card reader port

DWP005

Size: 3.75" X 3" X 0.65"

*Aluminum Case*



Brookstone®
www.Brookstone.com

Richard|Solo®
www.RichardSolo.com

Made for iPod | Works with iPhone | Mac | Universal

## References


"Address Book Programming Guide for Mac OS X," http://developer.apple.com/documentation/userexperience/Conceptual/AddressBook/AddressBook.pdf

"ABAddressBook Class Objective-C Reference," http://developer.apple.com/documentation/UserExperience/Reference/AddressBook/Classes/ABAddressBook_Class/ABAddressBook_Class.pdf

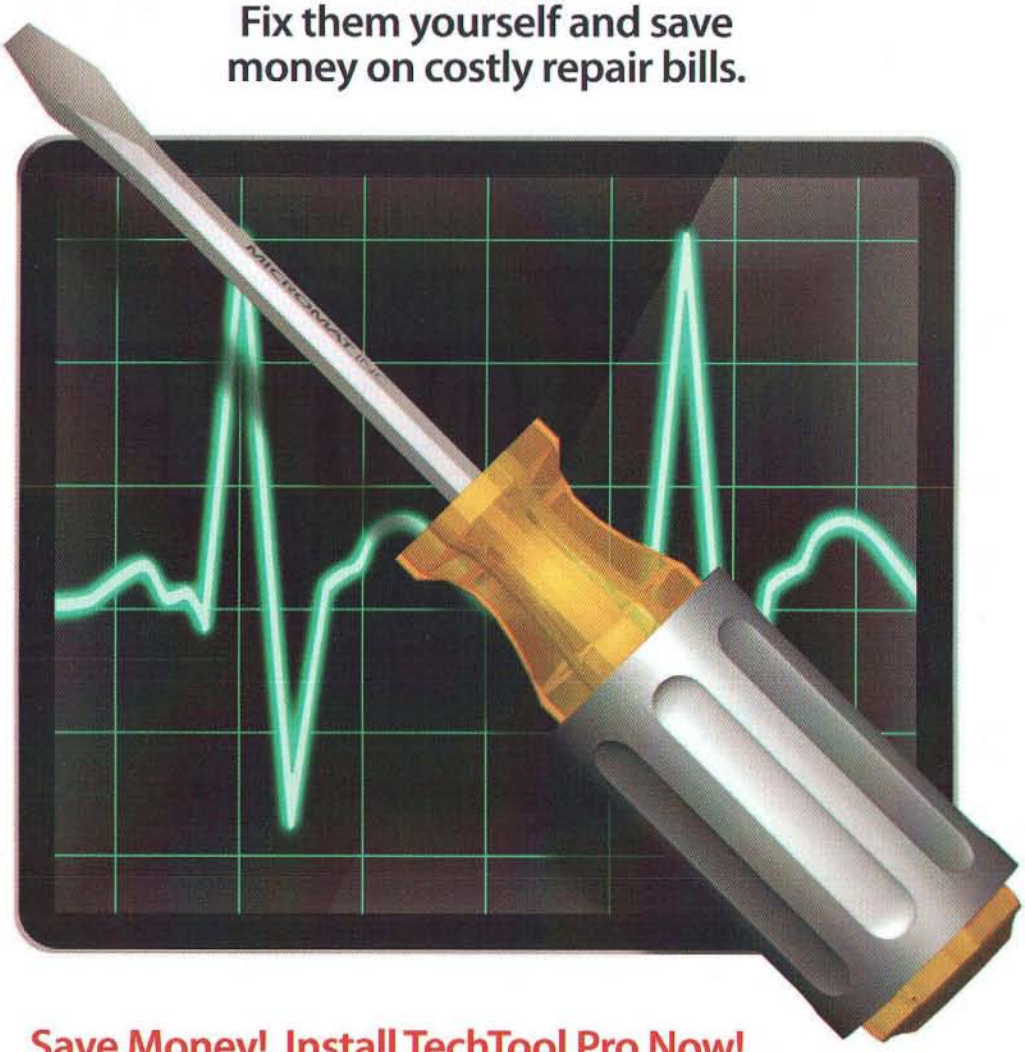
"ABPerson C Reference," http://developer.apple.com/documentation/UserExperience/Reference/AddressBook/C/ABPersonRef/ABPersonRef.pdf


## About The Author


***Ed Marczak is the Executive Editor of MacTech Magazine. He lives in New York with his wife, two daughters and various pets. He has been involved with technology since Atari sucked him in, and has followed Apple since the Apple I days. He spends his days on the Mac team at Google, and free time with his family and/or playing music. Ed is the author of the Apple Training Series book, "Advanced System Administration v10.5," and has written for MacTech since 2004.***

# TECHTOOL PRO 5

## Problems?
## Fix them yourself and save money on costly repair bills.



## Save Money! Install TechTool Pro Now!

- Keep your computer running smoothly.
- Help prevent problems in the future.
- Fix most of your problems on the spot.


Mac Universal

Micromat, Inc., 5329 Skylane Blvd., Santa Rosa, CA 95403, USA
800-829-6227 707-566-3831 info@micromat.com www.micromat.com

AcademicSuperstore.com campustech.com frys.com drbott.com bestbuy.com navarre.com macmall.com store.apple.com microcenterorder.com

# Everything that can be invented has been invented

## All we know about starting an Apple business, from the idea, to product launch and beyond.

*by Michael Göbel and Oliver Pospisil, Inspired By Life*

## Inside Inspired By Life

In 2006, I started to turn my idea into a software prototype. The feedback I got for the prototype from most people was: "Interesting idea. Is it based on academic research? Because the GUI sure looks like it."

It became clear that a GUI expert is needed to turn the prototype into something people love to use. While searching on the Internet for a person like that, Michael's website popped up, and after testing one of his applications, I called him right away.

After the first call, Michael was not convinced that my idea was worth investing any time in, especially when I told him that my financial resources are very limited. However, he at least took the time to check my prototype out as well as a three-page description.

After one week, he changed his mind. His curiosity was stirred sufficiently and we scheduled a meeting. After shaking hands, we immediately started to talk about the idea in general and what it could look like in software. As soon as Michael began to pinpoint all of the weaknesses in my concept, I was thoroughly convinced that he's the right person for the job. Michael didn't care whether I liked what he had to say or if it damaged my ego. He was sold on the idea – no time to waste on a potentially damaged ego! We were looking for the right metaphor to turn the idea into software.

Michael: "Okay, Oliver here's a stack of index cards. Describe the concept behind your idea."

We toyed around with the index cards for more than three hours.

Oliver: "Why don't we use the index card as the metaphor?"

Michael: "It seems promising, but we shouldn't stop here. Let's check some more options out on how we can do it."

Two days later Michael called me to tell me that the index card fits the best as a metaphor. It was just like arriving in the town called Eureka!

## Introduction

Charles H. Duell, commissioner of the United States Patent and Trademark Office in 1899, is allegedly the one who stated: "Everything possible that can be invented has been invented." 110 years later, you now have another chance to prove that the opposite holds true. To Mr. Duell's credit, rumor has it that he was not the one who made this statement after all. It seems to be a patently false, modern myth. However, your chance is not just a myth!

If the reason why you program Mac or iPhone applications is just for fun, it should not matter what other people think about your idea or how many download or buy it. Just go ahead and do it!

However, the situation changes when you want to make a living out of it, besides just having fun. What you need now is an application that people are willing to pay for – and not just once but also on a regular basis. First and foremost, you must have a terrific idea that you can turn into a beautifully crafted program.

In this article, we will tell you everything we know about how to stimulate curiosity, how to test an idea to find out whether or not it truly is a terrific one, why you should build up a good reputation at an early stage and what consequences you will be up against once you have decided to turn your terrific idea into a beautifully crafted Mac application.

## How to come up with an idea

After people come up with an idea and you ask them where they got it, they either say, "I don't know" or they try to rationally explain how straight their line of thought was. Personally, I don't really believe in all rational, straightforward explanations, instead, I go with the first one – "I don't know." To us, an idea is a destination towards which you travel.

### From curiosity to Eureka and back!

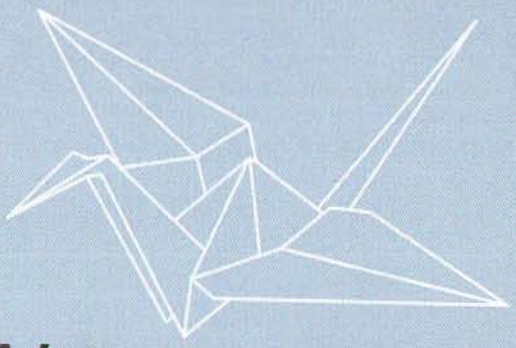
All you need to do is to start the journey inside of you. The very first stimulus for your mind is curiosity; the destination is Eureka (Greek: "I have found it") = your idea.

version 3.1

Snow Leopard Ready

**New With Version 3.1**

Full support for Gestures on new MacBooks with glass trackpads. Tested and operational on Snow Leopard. Dual Rep Tiffs for resolution independent graphics on Snow Leopard. WebColorPicker - select a color, paste a code snippet, must have for web, OS X application, and IPhone developers.



# eazydraw

have fun drawing on OS X

- Illustrations
- Web Graphics
- Technical Drawings
- Charts & Diagrams
- Text Layout

eazydraw.com

**Key Features Include:**

| | | |
|---|---|---|
| Layers | Scaled Drawings | Bezier Curves |
| Free Transform | Dashed Lines | Numerous Graphical Shapes |
| Arrows | Blends | Text Typesetting |
| Shadows | Connectors | Text Outline |
| Gradients | Dimensions | Text on Curves |
| Patterns | Transparency | Photo Support |
| Grids & Guides | Walls & Ribbons | and much more... |

Eazy Draw®

Getting to Eureka is not a one-way path, but rather, multiple paths. You do not even have to choose between them. Since it mainly takes shape inside of your mind, let it decide on the best path itself.

Here is our way to get from curiosity to Eureka:

I read a lot of books: How to start a start-up up, How to create mind maps, City guide of London, Ink heart, just to name a few. Whenever I come across something that I don't fully understand and I cannot find a satisfying answer quickly on the web, I go to Amazon and search for a book. When reading the book, I ask myself "What would this look like in software?" That's curiosity to me.

Michael takes a slightly different approach. He focuses on the reality of daily life and listens to his customers carefully. He asks himself how software can make life better or at least make it more fun. He's a technophile, just like you. Here's a good example: While we were walking around London he said: "Wouldn't it be great to bring a digital city guide for London along with me? What could a digital version do for me that a printed version can't? In combination with a 3G iPhone and Google Maps it could show me the shortest way to the Apple Store on Regent Street." – curiosity and Eureka! (Unfortunately, Michael and I didn't have a guide like that so we took the most popular route, the result of which was aching, blistered feet). The guys from Presselite demonstrated the same curiosity and Eureka!

Quite often, we ask ourselves "What would it look like in software?", before we get to a town like Eureka – an idea into which time and money are well worth dedicating. Any other questions like: "Who will buy it and at what price?" or "How much money will I make?" are simply secondary.

You still think you do not have this type of curiosity? Well, yes you do, because you opted to use OS X.

### The bottom line

Until your software application actually goes to market, you will spend many days filled with curiosity and Eureka. The one thing that is absolutely essential is curiosity – the first stimulus for your mind. Once you have it, let your mind go wild to get to Eureka – your idea.

Now let's see if your idea passes the terrific-idea-test.

## Terrific-Idea-Test

If all you really care about is making money, program a game, for example, an ego shooter and call it a strategy game. If what you want to do is to make this world a better place or to help others to do so, just test your idea.

### Test 1: Does your idea stick?

You have a lot of ideas night and day, don't you? Some of them come back the next day and again and again. These ideas are sticky. These are the ideas that are worth delving into and that are worth spending the time on to explore them.

Free Trial Available

June Special
Get
$50 off
Use code:
MTECH50



# Komodo IDE 5.1

## Code smarter and faster with ActiveState's award-winning professional development environment.

"If you work in any of the languages Komodo supports, you owe it to yourself to examine it (for free). If you work in any two of them, you probably should just buy it."

*— Andrew Binstock , Binstock on Software*

### Whatever Your Language

Komodo IDE supports PHP, Python, Ruby, Perl and Tcl, plus JavaScript, CSS, HTML and template languages like RHTML, Template-Toolkit, HTML-Smarty and Django.

### Whatever Your Platform

Mac? Linux? Windows? Yes, yes and yes! Buy just one license and use it on all your platforms.

### Work Smarter, Not Harder

Komodo IDE's award-winning feature set includes standard editor functionality, syntax checking and coloring, a regex debugging tool and more. And it's extensible, so hack away.

### Share the Workload

Team development is faster with source code control integration, a project manager and multi-user support. And when you buy four licenses, we'll throw in the fifth for free.

### Get Expert Help

You can rely on high quality support and an active user community for help whenever you need it.

Download your free 21-day trial now: **www.activestate.com/mactech**

**ActiveState**

The Dynamic Languages Company

Today, we realize that the stickiness of our idea has been growing over time. It was not that strong from the start. So, don't worry if the stickiness feels like a Post-It and not like Pattex. Your idea sticks, and that's what counts.

A sticky idea is also the source of energy that you need to sustain up to the very moment when you ask yourself "Does it all make sense and will it pay off?"

### Test 2: Does your idea solve a pain in the neck?

Who will no longer have a pain in the neck once they start using your software? How would it make the world a better place? Go out and find at least one person who will enjoy using your software and preferably who is willing to pay for it, too.

### Test 3: Does your better half tell you: "Please stop talking about it!"

Others are often better in seeing what we really do with our time. You are 100% certain that you've found something that is worth investing more time in when your better half begs you to stop talking about it – at least for one night!

When that happens, just give her or him a big smile, enjoy the evening together and the next day, get back to your idea as soon as possible.

### Test 4: Do others, not only friends, line up?

The first people you talk to with about your idea are most likely your better half, closest friends and family. Resist the tendency to trust their feedback—be it positive or negative. They really, truly care about you and that's the problem: They care so much that they want to do all they can to make sure that your endeavors do not fail.

So, just go out and ask people you have not met yet at parties, on the street, in pubs, restaurants – in fact – practically anywhere.

One important rule: Tell others what you want to do; don't tell them how you're doing it – that is your own magic and secret ingredient.

### Test 5: Do experts' eyes light up?

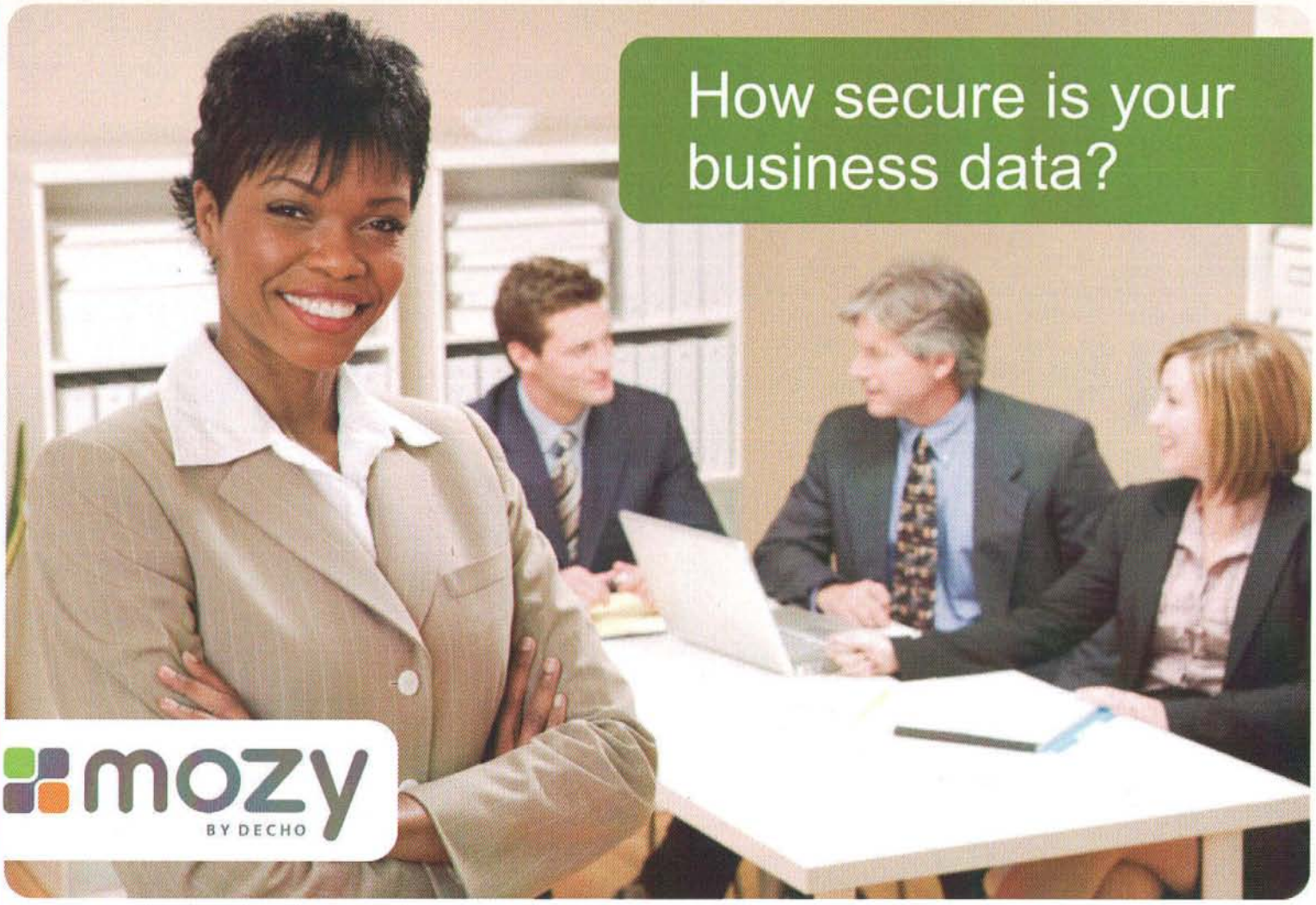
Maybe you came up with an idea after reading a book. Ask the author what he or she thinks about your idea. Ask them if they could recommend someone else to talk to. Search on the Internet for other experts and give them a call.

One important side effect: You're building up a reputation.

### Test 6: Do others design solutions for the same problem but not in the same way?

I assume that your idea is at least in one way comparable with ours: It is not totally new (new like the very first Internet search engine). Therefore, others offer a solution for the same problem like you. And you want to provide a different solution. That's great: Do they earn money with it? If they are earning money, there is a market, and a market is what you need.

---




## Protect your business with MozyPro online backup

MozyPro is the simple and safe way to protect all the important files on your business computers. A copy of all your files is stored offsite in secure data centers, so you are always covered in the event of file corruption, accidental deletion, hardware failure or even natural disaster.

Why take the chance? Get started today and save! Visit us online at **www.mozy.com/mactech** or call your industry representative at **877.669.9776** and receive 10% off your initial purchase when you use the promo code **MACTECH10**.

## Simple. Secure. Affordable.

- Mac and Windows support
- Online account management
- Multiple restore options
- The highest levels of security
- Backup solutions starting at less than $5

"MozyPro is the first online backup service I'm willing to offer my clients. No other company can offer as great of a service at such a great price."

**- David O'Connell**
OConnell I.T.

### Test 7 (optional): Will someone steal your idea?

Besides getting the core feedback, when we talk to other people about our idea we are often asked whether we worry about someone stealing our idea.

Our answer is: "Yes, this is a risk. But it is a risk we are happy to take, because otherwise we would never get valuable feedback at an early stage. If someone steals our idea, of course we will be very upset, however, we still know that we can rely on our own idea. We very much doubt that someone will steal your idea after a five-minute discussion. They do not know the whole story and they don't live and breathe your idea."

Concerning probability: Idea thieves will most likely wait until your idea is a successful product before they come up with their own, copycat solution.

### The bottom line

How many times did you answer with a "Yes"? – The more the better!

Be aware that a terrific idea does not necessarily translate into a mega success in terms of profit. However, it does mean that you're investing time and money in an area that promises potential success. This is the best thing you can hope for at this stage of your endeavors. You have finally made it to Eureka

## Reputation, an invaluable feature

Just imagine that your goal is not to develop a software application. Instead, you want to create a new hammer. And now, you've finally convinced the manager of the do-it-yourself shop in your neighborhood to display your new hammer on the shelves. You've designed an advertising poster with the slogan "New and revolutionary hammer!" How many carpenters do you think will buy it? If you get lucky, maybe a few. Like everyone else, carpenters are loyal, they stick to the things that they're used to and continue to buy the same hammer they always buy.

So how can we convince carpenters to buy the new (totally unknown) hammer? Either we need is a high-price "Think Different" advertising campaign or just one expert. Bring an expert on board who knows how to use your hammer, who likes it and who can hammer better results. Now imagine that the expert recommends your hammer to the editor of the carpenter tools magazine, they review your hammer and give it five nails. The next time you hear from the do-it-yourself shop manager, he will most likely tell you to deliver more hammers as soon as possible.

A good reputation is a feature that you cannot buy – some try to fake it, but it doesn't pay off in the long run. No genuine expert will accept and buy into a fake reputation (experts do know each other).

A good reputation opens up doors to the people who are of utmost value for the success of your idea. Talk to experts and ask for their advice and help. You'd be amazed at how many people are willing and eager to support a young start-up. Don't be surprised if the expert calls you and starts with "I thought about your idea, it's promising. I have an idea…".

One consequence of having a good reputation always holds true: The better you get, the better you better get.

press non-existing button to plan rand ( ) national anthem. thankyouverymuch

# work it does?

## Help us find out

Go here:
http://www.mellel.com/mactech.html

**Get 50% discount**


*Mellel*

## Personal consequences

My better half has been supporting me since 2006. Right from the very start, I have always been telling her: "Just hang in there for three more months, darling." But three years later, she is now skeptical yet still backing me all the way. Why? The answer is actually easy – she committed herself to support my original idea and she does see progress. I talked to her about my idea and she agreed that it's definitely worth a try. Her precondition: "Keep on working with your full-time employer until your start-up business pays off and, in return, you support my endeavors." I agreed, happily.

Personally, I would never have believed that it takes up so much time and gobbles up practically all of my financial resources in order to turn my idea into a real product. But it works out! You have to work before and after your full-time job, you dedicate your weekends, a lot of leisure time and even vacations to your terrific idea.

How do you think the ones we truly care about feel when they just see us sitting in front of the computer for endless hours, days and nights, and all of that just in the hope that one day in the future (that could be in a few years or in the worst case scenario even never) that you bring the same money home that you do today with your nine-to-five job? That's why you must reserve quality time for the people you truly care about. You have to find the right balance between personal life and work! Remember, you need time to relax, too. Your endeavor is a marathon, and not a sprint.

### The bottom line

If you don't want to destroy your love life or the steadfast, reliable relationships with your closest friends and family, then just get them onto the bandwagon! Just talk to them about your dreams and what they will get out of them. Get their commitment – and in my opinion, getting it from your better half is by far the most important one! Show them how you are making progress.

One positive side effect: They always know where you are – right in front of your computer.

## What's next?

It's easy to be curious all day long. Some day your curiosity will lead to Eureka, your terrific idea. Invest time in this sticky idea. Get your better half and the people you truly care about on board. They give you the energy you need when times get tough.

To make a living out of it, it is important to the take the business side into account as well. In the next article, we will talk about the business: The plan, the figures and the fun.

## Connect with us!

We want to share stimulating, innovative ideas with you and we really look forward to your feedback! Is anything missing or do you think something could be fleshed out in further detail? Just let us know and write to oliver.pospisil@inspiredbylife.com.

## Bibliography and References


### Books:

Walsh, Bob. *Micro-ISV: From Vision to Reality*. New York, 2006.
Heath, Chip & Dan. *Made to Stick*. New York, 2008.

### Websites:

- Charles Duell: http://en.wikipedia.org/wiki/Charles_Duell
- Eureka: http://en.wikipedia.org/wiki/Eureka_(word)
- Pattex: http://www.pattex.com
- London City guide by Presselite: http://www.londontubeiphone.com/
- http://www.mac-developer-network.com
- http://www.47hats.com
- http://www.ericsink.com




## About The Authors

***Michael started MOApp up in 2004 and he has now developed more than ten applications - six of them are Apple staff picks. He does everything from software development, icon design, website development to sales management and public relations.***

***Oliver has been in the software business for over ten years, specializing in areas ranging from Palm programming to large-scale, in-house Java projects. In 2006, an idea grabbed his attention that both are now working on. He is still working full-time for a German retail company and will be until the new business starts paying off their bills.***

# Email Marketing Made Easy... Just Like the Mac!


**The standard in permission-based email marketing.**

- Grow your contact lists
- Send quality emails
- Track your responses
- Survey your customers

Plans starting at only $9.99 per month

## Sign up for a FREE 30 Day Trial Today!

## www.BenchmarkEmail.com

benchmark email

**800.430.4095**

# COMMUNICATION IN PROJECT MANAGEMENT

www.ConceptDraw.com

Communication is the cornerstone of effective project management. This is a principle that is clearly understood by most project management participants. Many times the most difficult portion of a project is effective communication of project objectives, goals, and desired outcomes of the project to team members.

The more resources involved in attaining project goals, the more complex communication tasks become. ConceptDraw Office is able to clearly define and implement the communication processes for any project.

With ConceptDraw Office, you can easily build long-lasting customer relationships, expedite project activities, and keep control over projects by delivering necessary information to responsible parties in the corresponding deadlines and in appropriate formats.

As 90% of a project manager's time is invested in communication, it is critical that there is a tool in place that best utilizes that time. A large portion of that 90% is invested in daily communications, such as, project monitoring, coaching team members, modifying project details, and responding to information requests.

Incorporating an effective communication strategy is a core component to a project management team, to keep projects in line with the expectations of a company. Once a project strays off target it is very tedious to get it back on track. Many times project results can become mired in mediocrity because clear channels of communication were not properly setup from the very beginning.

When implemented properly, a good communication process positively transforms the entire project experience for all participants. ConceptDraw Office addresses communication as a critical component that is vital to managing communication throughout the entire project life cycle.

There are five main components and stages of a project life cycle:

1. Generate Ideas
2. Plan Actions
3. Organize the Project
4. Communicate Strategy to the Team
5. Manage Project Execution

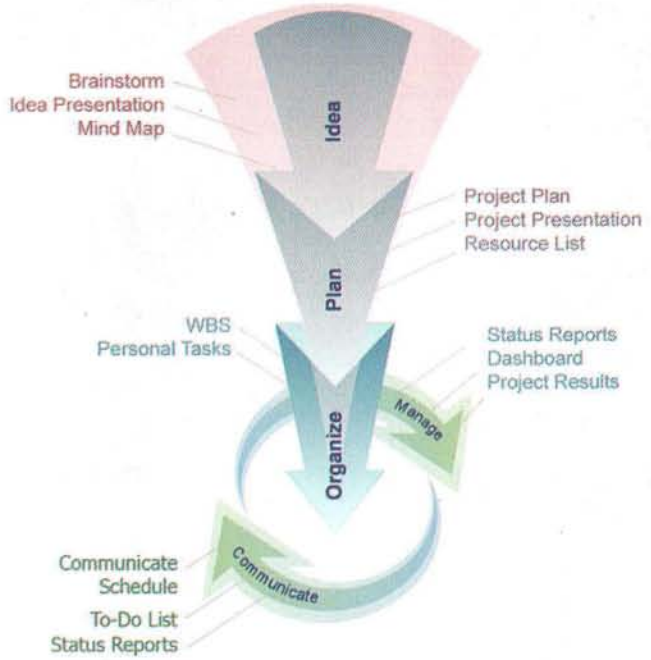



***Figure 1.*** *Components of a Project Life Cycle.*

Before a new project reaches implementation, there is a necessary stage of research, discussion, and creativity. This Idea Stage is where project ideas are generated, selected, discussed and finalized.

The main issue facing project managers at this stage is the need to understand the main goal and its requirements. Whether you are working on an internal or external project, there is always a need to discuss ideas and potential strategies, as project resource requirements must be calculated and understood amongst all project participants.

Mind maps are often used to present a combination of ideas in a readily understood visual format. ConceptDraw MINDMAP helps map out and organize ideas in this initial stage, in an efficient manner that is appropriate for any project management workflow. A mind map is an effective way to visually capture and present concepts, build strategy maps and assemble strategic action plans. The intuitive user interface for brainstorming is an immeasurable value to any project.

ConceptDraw MINDMAP also makes it simple to consolidate generated ideas and produce professional looking organizational diagrams. The suggested forms of communication at the Idea stage are brainstorm sessions, mind maps, business process models, and presentations.

The **Plan Action Stage** details the ideas, plans out the actions necessary for implementation, and forecasts the expected results of a project to higher management and investors. Precise organization and planning when creating draft timelines and budget are critical components for future projects. At the **Plan Action** and **Organize Project Stages**, a project timeline is mapped out for task and resource (people, equipment, and materials) allocation. ConceptDraw Office contains the tools necessary to handle these two important stages.

ConceptDraw PROJECT is well integrated with ConceptDraw MINDMAP, saving lots of time when setting up a project and all the other necessary components. The mind map created with ConceptDraw MINDMAP during the Idea stage, can easily be transformed into a project Gantt chart, simplifying job delegation and describing responsibilities.

ConceptDraw PROJECT is also tightly integrated with ConceptDraw PRO, requiring just one click to automatically build a Work Breakdown Structure (WBS) chart for your project. Now it is easy to keep your WBS up to date. If more changes are needed, modifications to the project can be directly made in ConceptDraw Project with the automatic rebuild of associated WBS.

ConceptDraw PROJECT also makes it easy to send detailed To-Do lists to all project participants. The typical amount of time it takes to communicate these tasks is significantly reduced to a minimum.

Suggested forms of communication for the Plan Actions and Organize Project stages are Gantt charts, WBS charts, flowcharts, Network diagrams, and To-Do lists.

Two other major components necessary for the success of a project are **Daily** and **Management Communication**. Having a communication management strategy that is supported by ConceptDraw Office helps detail the needs and expectations of the team, how project data will be exchanged, and the responsibilities of each team member. Creating professional and impressive presentations of project data helps build company credibility with customers and stakeholders.

The communication process continues as long as a project is in action, to keep information flowing between all members of a project.

Information flow is vital to a project, as it presents assigned tasks, reports results, and presents decision making data to project participants. Information flow connects all participants of a project, providing necessary information in a timely manner and in a format that is most appropriate for the tasks.

ConceptDraw Office minimizes the amount of project rework necessary and improves the everyday workflow of any project by utilizing visual methods to represent pertinent information and business processes. With visual methods it is easy to:

- open project files in ConceptDraw MINDMAP to modify schedules and project details
- effectively determine workload-to-work capacity and resource allocation in ConceptDraw PROJECT
- present project status to clients and colleagues in the form of Report Maps or by utilizing ConceptDraw MINDMAP features such as Presentation mode and export to MS PowerPoint
- build charts and diagrams in ConceptDraw PRO that document processes and reports on project status

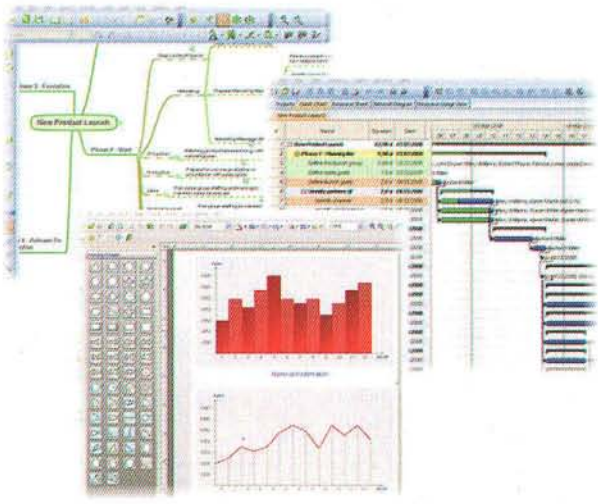


***Figure2.*** *Visual Representation on Project Information.*

During a project life cycle dozens of documents are created for different purposes. ConceptDraw MINDMAP makes it convenient to keep all the related files to a project in one place. A single Knowledge Map, supporting multiple pages and hyperlinks, easily becomes the main information source for a project of any size.

Suggested communication formats for the Communicate and Manage stages are Report Maps, Resource Usage view, Visual Reports, Project Dashboard, Knowledge maps, mind maps, and PowerPoint presentations.

Clear communication within a company helps build a collaborative work environment, with an emphasis on a shared vision of goals. ConceptDraw Office offers an effective and innovative approach to managing the five main components and stages of a project life cycle.

Learn more about ConceptDraw Office at:
**www.conceptdraw.com**

# Simple Call Stack Logging

## Who called NSLog()?

by Sengan Baring-Gould

In this article I present an extension to `NSLog()` which not only prints out a user specified message but also lists the function calls that led to its invocation. By the end of this article you will have a new tool you can use in your own applications, and you'll understand how it works so that you can adapt it to your needs.

## Why log?

`NSLog()` is an important tool for debugging. It can be placed anywhere in an application to log internal state.

Many programmers prefer logging to using a debugger, as it helps them concentrate on possible causes of a problem while filtering out irrelevant information. Logging provides a complete textual record of the problem that can be studied later.

Debuggers on the other hand interrupt the developer who must record by hand all the relevant state before letting the application continue. Continuing is an unforgiving operation: once performed, prior state that was not recorded is lost.

Brian W Kernighan (one of the authors of the seminal text on the C language) said:

*"As a personal choice, we tend not to use debuggers beyond getting a stack trace or the value of a variable or two. One reason is that it is easy to get lost in details of complicated data structures and control flow; we find stepping through a program less productive than thinking harder and adding output statements and self-checking code at critical places. Clicking over statements takes longer than scanning the output of judiciously-placed displays. It takes less time to decide where to put print statements than to single-step to the critical section of code, even assuming we know where that is. More important, debugging statements stay with the program; debugger sessions are transient".*

However, if `NSLog()` is invoked from a function that is called from many other functions, determining which function call caused the bug can be very difficult. We need a record of the function calls that led to the invocation of `NSLog()`.

## Where we are headed

Our new debug function `debugLog()` will print out any message we want the same way NSLog() does. Following the message, it will list the function invocations that led it to be called (see Figure 1, below):

The first line follows `NSLog()`'s traditional format: the date, the time, the name of the application (TestDebugLog) and then the message we passed as argument: "C++ Constructor".

On the next lines, `debugLog()` lists the function invocations that led it to be called:

- `debugLog()` was called from the C++ constructor `CPP::CPP()` defined in TestDebugLog.
- `CPP:CPP()` was called by `main` also defined in TestDebugLog.
- `main` was called by `start` also defined in TestDebugLog. (`start` is the function the Operating System calls when it starts an application).

The module name between parentheses specifies where a function is defined. In the following example, `NSPopAutoreleasePool` is shown to be defined in the Foundation framework (see Figure 2, below):

## Obtaining the list of function invocations

Obtaining the list of function invocations to print is a two step process. First, `debugLog()` must obtain the addresses of the functions that called it. A computer uses addresses to

```
2009-04-05 15:37:58.119 TestDebugLog[14442:10b] C++ constructor
        0000301f - CPP::CPP() + 33                (TestDebugLog)
        0000316c - main + 86                      (TestDebugLog)
        00001a5e - start + 54                     (TestDebugLog)
```

**Figure 1**


## Why would you want *altered* fidelity?

The award-winning Etymotic hf2 headset delivers the most accurate fidelity of any earphones available today. We constantly test against the competition. The uncompromising performance standards and noise-excluding eartips provide a level of clarity that lets you hear parts of your music you may have never heard before.

*Experience unaltered high fidelity. You will be amazed.*

ETYMŌTIC
*The science of pure music*

**For more information on how we measure fidelity or to see our full line of high fidelity earphones and headsets visit www.bestipodsound.com**

```
2009-04-05 15:40:03.921 TestDebugLog[14462:10b] Objective-C dealloc
        000030d3 - [Objc dealloc]  + 33          (TestDebugLog)
        91117e4f - NSPopAutoreleasePool + 431   (Foundation)
        000031d1 - main + 207                    (TestDebugLog)
        00001a4a - start + 54                    (TestDebugLog)
```

keep track of what it is doing, but addresses are not specified in a program's source-code.

Then `debugLog()` must translate these addresses into the function names that appear in the program's source code. Three different methods are required to obtain C, C++, and Objective-C function names.

## 1. Retrieving the list of called functions' addresses

Compilers transform source-code into machine code that computers understand. When a function is called, the caller's return address must be saved so that the CPU can continue running the caller after the function invocation completes. In the context of this discussion, we will assume that these return addresses are stored on the stack. (We will ignore specific optimizations used by the PowerPC and x86 CPUs).

Unfortunately, the stack also records other information, such as local variables and function arguments. Determining precisely which items in the stack are return addresses requires compiler specific knowledge. Conveniently, the new version of gcc which ships with Leopard provides a new function, `backtrace()`, which gives us the return addresses in the current stack.

Remember that inlined functions are embedded within their callers rather than being invoked. This means backtrace will not see them in the stack and they will not be listed by `debugLog()`.

## 2. Function layout in memory

Functions are compiled independently by the compiler and occupy contiguous areas of memory. Therefore if we know the starting addresses of any two consecutive functions f and g, and if we have an address x which falls between f and g ( $\&f \le x < \&g$ ), we know that x belongs to the function f.

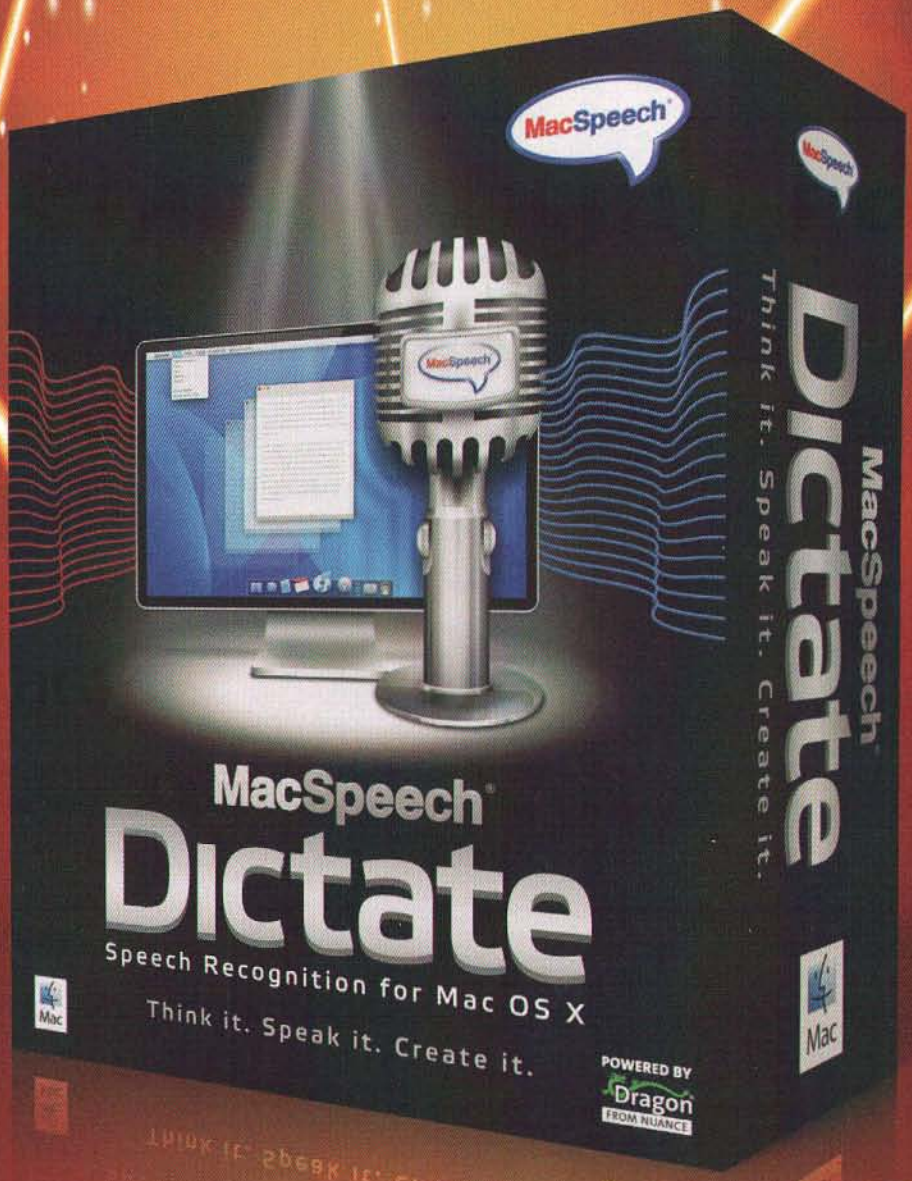
One rarely has addresses that match the beginning of each function precisely. Therefore function-lookup functions are designed to return information about the preceding function when given an address. Our case matches this scenario: the return addresses provided by backtrace occur within functions. Thus we can safely ignore the difference between return addresses and starting function addresses for most of this discussion. Similarly, we'll adopt the standard convention of referring to starting function addresses as function addresses.

# Better. Stronger. Faster.

Better accuracy. Stronger recognition. Faster performance.

**Introducing MacSpeech Dictate 1.5**

Premier speech recognition for the Mac.




## www.macspeech.com

Available from MacSpeech, Apple, and other fine Macintosh retailers.
Visit the MacSpeech website for a complete retailer listing.

MacSpeech

### 3. Retrieving C function names

When an application is first loaded into memory, it needs to be told the addresses of the external library functions it wants to use. Because libraries are updated independently of applications, the addresses of their functions may change, although the names of their functions will not. It is the responsibility of the dynamic linker to give each application this information.

The dynamic linker reads function names from a symbol table embedded in the application and the libraries the application uses. The symbol table lists all the C function addresses and the C function names that can be accessed externally. Therefore if we know an address, we can ask the dynamic linker for the corresponding function name. The `backtrace_symbols()` function provides this functionality.

Because the dynamic linker only knows about externally visible function names, `backtrace_symbols()` always returns the nearest preceding external symbol. Static C functions are not exported and will not be given the correct name by the linker. Most symbols will be exported as external if you compile your application in Debug Configuration. This is not true if you compile it in Release Configuration. The UNIX utility `nm` lists exported function names with a preceding capital `T` letting you check if an unexpected function name shows up.

### 4. Retrieving Objective-C method names

Objective-C does not use the dynamic linker. Instead it uses the Objective-C runtime, which like the dynamic linker keeps track of all known method addresses and names. (A method is a function defined within a class). However there is no equivalent to `backtrace_symbols()` which returns a function name when given a function address. We must build one ourselves.

The Objective-C runtime provides a function to enumerate all the Objective-C classes that can be invoked by the application, including those in the frameworks bound to the application. It also provides a function to enumerate the methods in any Objective-C class. With these components we can obtain every method's address and name.

## Implementation

This code is Objective-C++ so don't forget to use the ".mm" extension for your implementation filename. We start with the necessary includes:

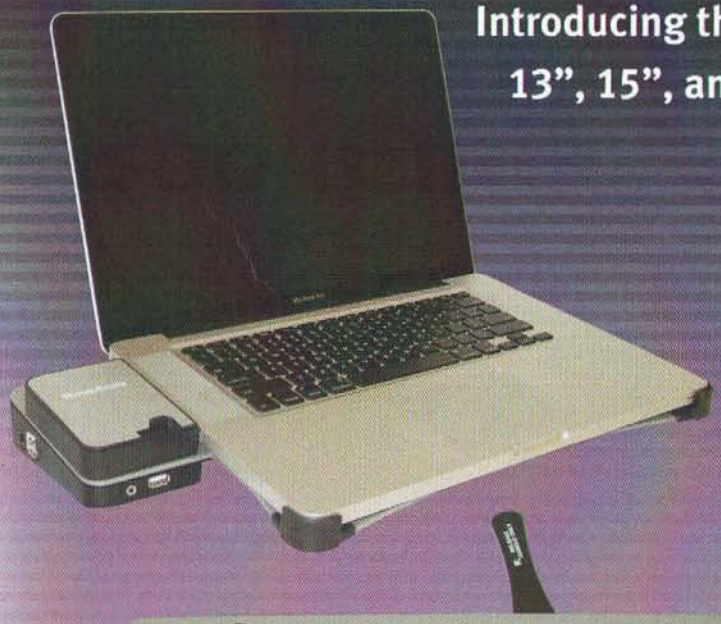
***DebugLog Implementation File: SBGDebug.mm***

```
#include "SBGDebug.h"

#include <cxxabi.h>
#include <map>
#include <string>
#include <execinfo.h>
#include <stdio.h>

#import <objc/Object.h>
```

# BookEndz® Docking Stations for Apple Computers

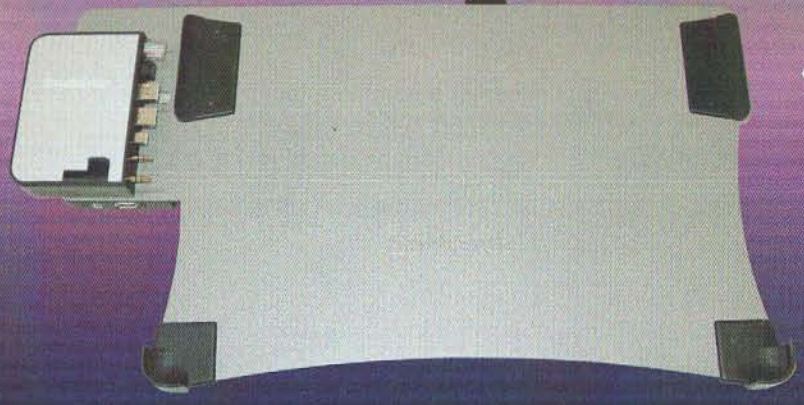
## DOCKING STATIONS FOR APPLE COMPUTERS

**Convert your MacBook Pro® or MacBook®
into a desktop in seconds
without misplacing cables or damaging connectors.**



**Introducing the New Docking Stations for the 13", 15", and 17" Unibody Apple Laptops**

- Aluminum Plate helps in cooling of notebook
- Connectors are routed to back of Docking Station
- 5 USB 2.0 compliant ports (4 port powered or unpowered hub, AC/DC adapter included)
- Gigabit Ethernet RJ45
- Convenient front mounted Headphone and USB jack
- Security Lock Accessible



**Visit our website for latest product announcement www.BookEndzdocks.com**

BookEndz®
Manufactured by Olympic Controls
1250 Crispin Drive • Elgin, Illinois 60123 • USA
Phone: 847-742-3566 • Fax: 847-742-5686 • Toll Free: 888-622-1199
E-mail: Sales@BookEndzdocks.com

## 1. Finding Objective-C methods

We want to build a method-lookup function that will return the method in which an address lies. We know how to enumerate the name and address of every Objective-C method available to the application, but this enumeration is expensive. Therefore we need a data structure in which to store the locations of the member functions.

The data structure must return the correct member function when queried with any address belonging to that member function. `NSDictionary` does not provide this functionality, but C++'s Standard Template Library's (STL) `map` container does.

`map::upper_bound` returns an iterator to the first element in the map whose key is larger than the queried key. For a map with method addresses as keys and method names as values, `map::upper_bound` will return an iterator pointing to the name of the method following the one we are looking for. Simply decrementing the iterator will make it point to the preceding method name.

```
// Lookup Function names

static std::map<uint32, std::string>* objectiveCMethodNames = NULL;

inline static uint32 lookupFunction(uint32 addr, const char** name)
{
    if (objectiveCMethodNames == NULL)
        return NULL;

    // Find the next function
    std::map<uint32, std::string>::iterator iter
        = objectiveCMethodNames->upper_bound( addr );

    // Go back a function: now we are looking at the right
one!
    --iter;

    *name = iter->second.c_str();
    return iter->first;
}
```

Populating the STL map is a simple matter of iterating through all the classes known to the Objective-C runtime, and enumerating their methods.

```
// Add classes by stepping through their method lists.

inline static void  addObjectiveCMethod(uint32 addr, const char* name)
{   (*objectiveCMethodNames)[addr] = std::string(name); }

void addClass(Class c)
{
    unsigned int method_count;
    Method *method_list = class_copyMethodList(c, &method_count);
    for (int i = 0; i < method_count; i++)
    {
        Method      func = method_list[i];
        const char* name = sel_getName( method_getName( func ) );
        uint32      addr = (uint32) method_getImplementation( func );
        addObjectiveCMethod(    addr,
                                [[NSStringstringWith¬
Format:@"[%s %s]",
                                class_getName(c), name]¬
cString]);
    }
}
```

# Amp Up Your Mac!™

## Call MacMall for our latest specials with your NEW Mac!


**6 Months Same as Cash!**
*Valid for purchases over $500. Call for details.*

**Up to $250 Cash Back!***
*On select Apple computers from our Web site at www.macmall.com. After mail-in rebate.*

**FREE Parallels Desktop!***
*After mail-in rebate with purchase of an Apple computer.*

### NEW 15" Aluminum MacBook Pro
2.4GHz with 2GB SDRAM and 250GB Hard Drive
FREE Parallels Desktop!* *Includes NEW iLife '09!*

~~$1994~~ - $150 mail-in rebate* = **$1844!*** #7684020
*After mail-in rebate.


**SAVE $150!**
**17" MacBook™ Pro**
2.66GHz with 4GB SDRAM, 320GB HD and SuperDrive
FREE Parallels Desktop!*
~~$2794~~ - $150 = **$2644!*** #7732924 *After mail-in rebate.


**SAVE $50!**
**NEW 20" iMac®**
2.66GHz with 2GB SDRAM, 320GB HD and SuperDrive
FREE Parallels Desktop!*
~~$1194~~ - $50 = **$1144!*** #7758229 *After mail-in rebate.


**SAVE $65!**
**13" MacBook™**
2GHz with 2GB SDRAM, 160GB HD and SuperDrive
FREE Parallels Desktop!*
~~$1294~~ - $65 = **$1229!*** #7684018 *After mail-in rebate.


**Apple® iWork™ '09**
*Word processing, spreadsheets and presentations that put the fun back into your work!*
only **$74!** #7732910

**LaCie 1TB d2 Quadra External HD**
*eSATA 3Gb/s, FireWire 800, FireWire 400 and USB 2.0!*
only **$174.94!** #7702561

Authorized Reseller

**MacMall**
*Your #1 Apple Superstore!*

Source code: MACTECH

## Call 1-877-233-2838 or visit macmall.com

*CASH BACK-Purchase select computer models from MacMall and receive up to $250 cash back via MacMall mail-in rebate. Ends 5/17/09. • FREE PARALLELS DESKTOP OFFER-Get Parallels Desktop 4.0 for Mac free after $20 mfr. and $60 MacMall mail-in rebates with purchase of any new Apple computer. Price before rebates is $80. Ends 5/17/09. • ALL OFFERS VALID WHILE SUPPLIES LAST. Download rebate coupons at www.macmall.com/rebates. For rebate terms and conditions, please visit our Web site and enter the applicable part number. Although we do our best to achieve 100% accuracy, occasionally errors and inaccuracies do occur. Should you encounter an error or inaccuracy, please inform us so it can be corrected.

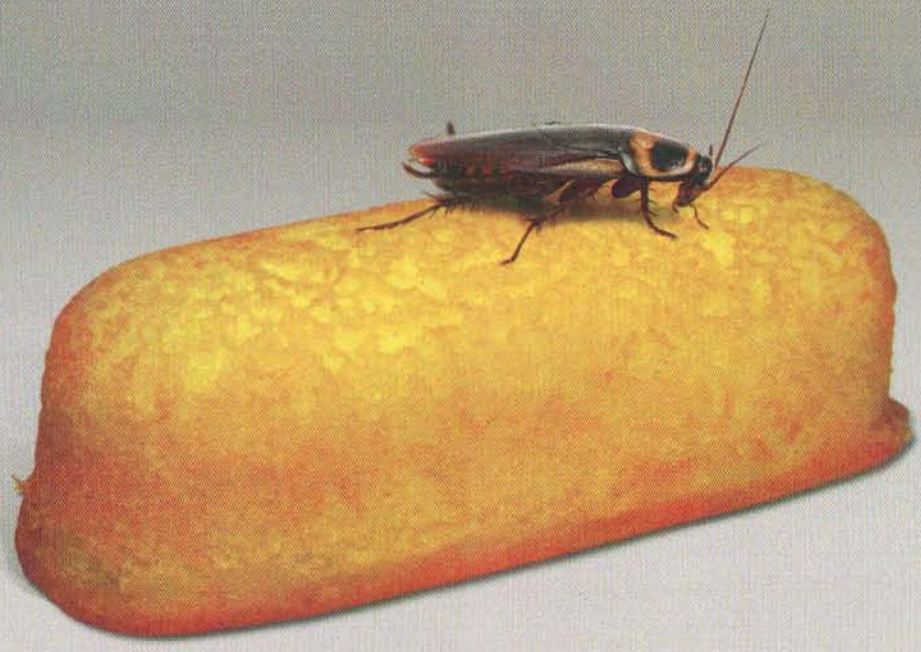
We could call the code that enumerates the Objective-C methods explicitly from main, but that requires remembering to add the call to each new application that uses `debugLog()`. Instead I can put the enumeration code into the load method of an Objective-C class which is guaranteed to be called if `debugLog()` is built into the application. The only gotcha is that the load method is invoked before Cocoa has created an `NSAutoreleasePool`. That's why the code in Figure 3 creates its own `NSAutoreleasePool` to avoid memory leaks.

```
@implementation SBGDebug

+ (void) load
{
    if (objectiveCMethodNames != NULL)
        return;

    NSAutoreleasePool* pool = [[NSAutoreleasePool alloc] init];
    objectiveCMethodNames = new std::map<uint32, std::string>();

    int numClasses = objc_getClassList(NULL, 0);

    if (numClasses > 0 )
    {
        Class *classes = (Class*) malloc(sizeof(Class) * numClasses);
        numClasses     = objc_getClassList(classes, numClasses);

        for (int i = 0; i < numClasses; ++i)
            addClass(classes[i]);

        free(classes);
    }

    [pool release];
}

@end
```

**Figure 3**

## 2. Deciding whether a function is C, C++ or Objective-C

The return addresses provided by `backtrace()` could belong to a C function, a C++ function or an Objective-C method. We need a way to decide which case we're dealing with. We start by asking both the C/C++ function-lookup function and the Objective-C method-lookup function to what function they believe a return address belongs. We will obtain two addresses *a* and *b* which should both be smaller than the return address *r*. Because functions are contiguous and do not intersect with each other, *a* and *b* must differ, and one of them must be lower than the other. For the same reason, the return address *r* cannot belong to the function with the lower address, as that function must end before the higher address. Therefore we use the name of the function that starts at the higher address (see Figure 4).



The only things that last longer than our committment to our customers.

**CELEBRATING 22 YEARS OF DEDICATION.**

SELL. MANAGE. PROTECT.

www.esellerate.net

The output of `backtrace_symbols()` is an array of C-strings. We use `sscanf()` to parse them. The resulting function names are passed to the C++ demangler to convert into human-readable form if they are C++ names.

`debugLog()` uses `printf()` rather than `NSLog()` to avoid printing the application's name at the beginning of each line.

`debugLog()` is declared as `extern "C"` so it can be linked directly to Objective-C code. The class interface file does the same:

***DebugLog Interface File: SBGDebug.h***

```
@interface SBGDebug : NSObject

+ (void) load;

@end

#ifdef __cplusplus
extern "C" {
#endif

void debugLog(NSString* format, ...);

#ifdef __cplusplus
};
#endif
```

# Using DebugLog

Invoke `debugLog()` just as you would `NSLog()`:

***Test file: TestDebugLog.mm***

```
#import "SBGDebug.h"

// C++ test

struct CPP      { CPP(); ~CPP(); };

CPP::CPP()      { debugLog(@"C++ Constructor"); };
CPP::~CPP() { debugLog(@"C++ Destructor"); };

// Objective C test

@interface Objc : NSObject
@end

@implementation Objc

- (id)      init
{ debugLog(@"Objective-C init"); return [super init];
};

(void)    dealloc
{ debugLog(@"Objective-C dealloc"); [super dealloc]; };

@end

// Main

int main(int argc, const char* argv[])
{
    NSAutoreleasePool* pool = [[NSAutoreleasePool alloc] init];

    CPP    cpp;
    Objc* objc = [[[Objc alloc] init] autorelease];
```

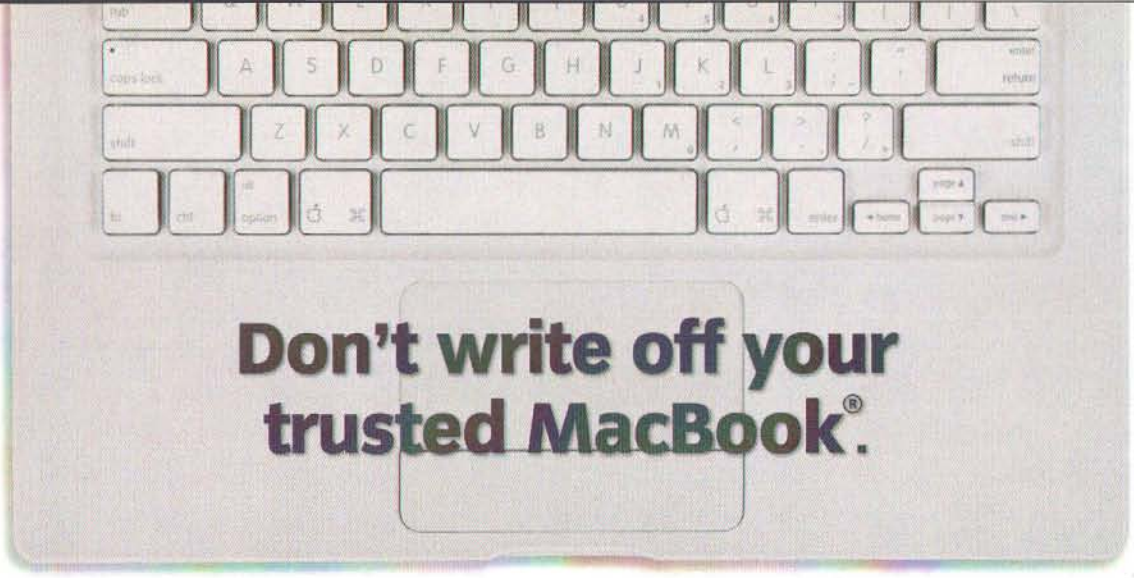
```
extern "C" void debugLog(NSString* format, ...)
{
    // Print the debug message

    va_list arguments;
    va_start(arguments,format);
    NSLogv(format, arguments);

    // Dump the callstack

    uint32 callstack[128];
    int  frames = backtrace((void**) callstack, 128);
    char** strs = backtrace_symbols((void**) callstack, frames);

    for (int i = 1; i < frames; ++i)
    {
        char functionSymbol[64*1024];
        char moduleName     [64*1024];
        int  offset         = 0;

        sscanf(strs[i], "%*d %s %*s %s %*s %d", &moduleName,
                                                &functionSymbol, &offset);
        uint32 addr = callstack[i];

        if (objectiveCMethodNames)
        {
            const char* objcName;
            uint32      objcAddr = lookupFunction(addr, &objcName);

            if (        (objcAddr != 0)
                    &&  (addr > objcAddr)
                    &&  (addr - objcAddr < offset))
            {   printf("\t%8.8x - %s  + %d\t\t(%s)\n", addr, objcName,
                                                        addr - objcAddr,
moduleName);
                continue; };
        };

        int   validCppName;
        char* functionName = abi::__cxa_demangle(functionSymbol, NULL, 0,

&validCppName);
        if (validCppName == 0)
            printf( "\t%8.8x - %s + %d\t\t(%s)\n",
                    addr, functionName, offset, moduleName);
        else
            printf( "\t%8.8x - %s + %d\t\t(%s)\n",
                    addr, functionSymbol, offset, moduleName);

        if (functionName)
            free(functionName);

    }
    free(strs);
}
```

**Figure 4**

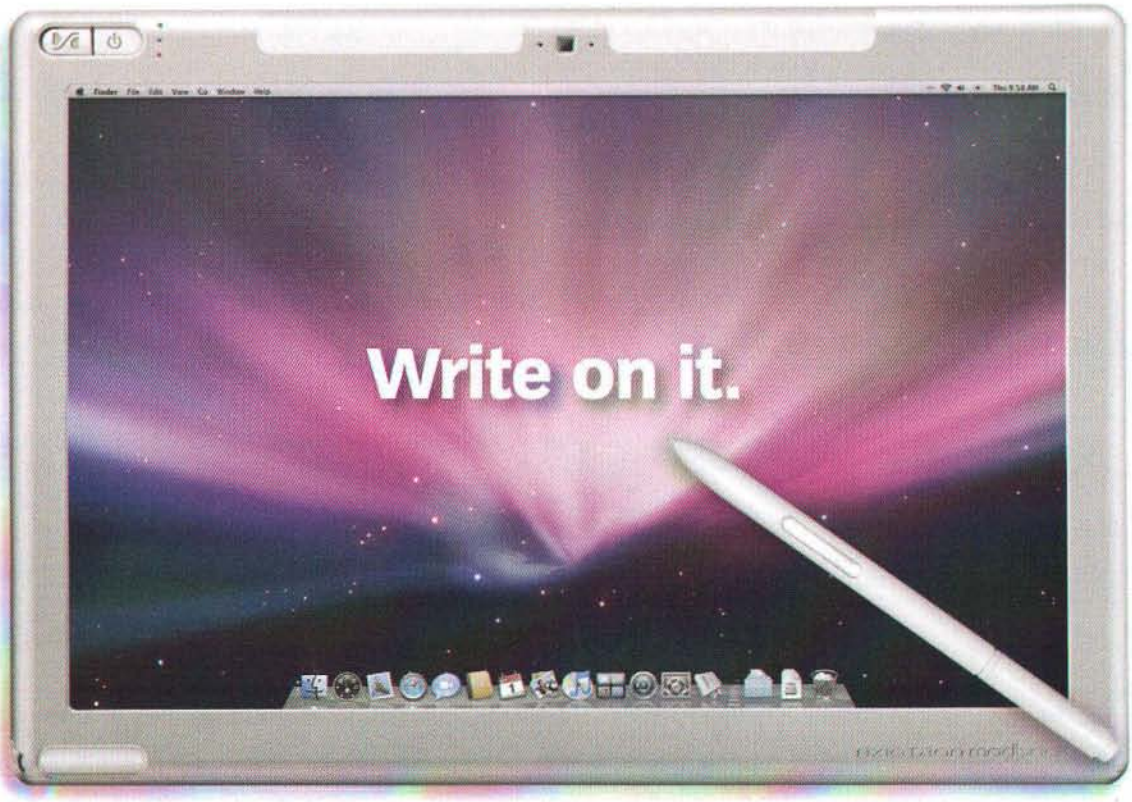
```
    debugLog(@"C - test arguments work too: %@", objc);
    [pool release];
    return 0;
}
```

# Conclusion

Leopard provides all the components necessary to build a cross-platform `NSLog()` which can print function call traces. Although the code I provide assumes compilation to a 32-bit executable, extending it to 64 bits should be straightforward as only standard library functions are used.



# Don't write off your trusted MacBook®.




## Introducing Axiotron® Modservice™ — transform your *existing* Apple® MacBook into an Axiotron Modbook®.

It works like this: Sign up for Modservice.* Choose your upgrade options and the warranty you want. Turn in your computer to one of our Axiotron Authorized Service Providers for conversion. Then kick back and enjoy your new Modbook. Draw, sketch and write by putting a pen to the screen of the best tablet computer with industry-leading Wacom® Penabled® technology and Mac OS® X. Portable and versatile, the Modbook empowers your creativity and imagination.

With Axiotron Modservice, your Apple MacBook computer gets revitalized, renewed and revolutionized — into a whole new product, the award-winning Axiotron Modbook. Visit www.axiotron.com/modservice for details.

*the one and only* AXIOTRON modbook

*Some service limitations and restrictions apply.

You can download the entire Objective-C++ project from the MacTech ftp source archive at ftp.mactech.com/src/mactech/volume25_2009/25.06.sit.

Don't forget to compile it and run it in Debug Configuration!

## References:

backtrace() : man backtrace.

backtrace_symbols() limitations : http://lists.apple.com/archives/darwin-dev/2009/Mar/msg00111.html.

backtrace_symbols() : man backtrace_symbols.

abi::__cxa_demangle : http://www.ib.cnea.gov.ar/~oop/biblio/libstdc++/namespaceabi.html.

Objective-C runtime: http://developer.apple.com/documentation/Cocoa/Reference/ObjCRuntimeRef/ObjCRuntimeRef.pdf.

The Standard Template Library: http://www.sgi.com/tech/stl/

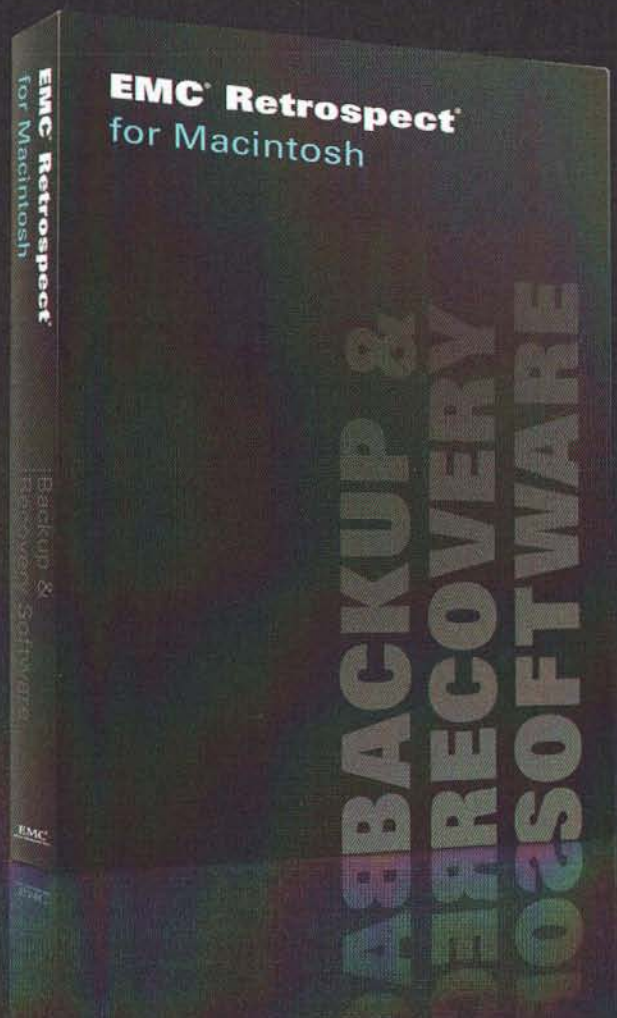
## About The Author

***Dr. Sengan Baring-Gould is a Boulder, Colorado-based independent Mac OS X developer and writer. He is available for consulting and specializes in Algorithms, AI, Cocoa, Debugging tools, High performance code, and UIs. He can be reached at sengan@ansemond.com.***

NEW!

# EMC® Retrospect® 8

## backup and recovery software for small and medium businesses




## The most trusted name in Mac backup

Now available for PowerPC & Intel Macs!

All-new EMC Retrospect 8 for Macintosh provides the reliability, ease of use, power, and flexibility you need to protect critical data on Mac and Windows PCs and servers. EMC Retrospect includes a state-of-the-art Mac user interface and enterprise-level features — including remote management of one or more backup servers, disk-to-disk-to-*anything* backups, Xsan support and custom reporting — at a fraction of the cost of other products.

Download a free 45-day trial at **www.retrospect.com/wwdc**

EMC²
where information lives®

# Demystifying PKI

## Part One in a Series of Articles and How-Tos about PKI technology in the OS X environment

By Michele (Mike) Hjörleifsson

## Introduction

Public Key Infrastructure, or PKI, is a mature set of tools and technologies that serves as the basis for securing most network communications and dozens of other security technologies. It is one of the most misunderstood technologies in the IT arena. This series of articles presents a brief history of PKI, explains how it's currently used, and describes how you can implement PKI in both small and large OS X implementations for various types of security without breaking the bank or causing excessive brain strain.

## What is PKI and Why Should I care ?

Let's start at the beginning,. PKI has evolved from a theory and paper published in 1976 by Diffie-Hellman describing the use of asymmetric ciphers versus symmetric ciphers in a white-pages-like directory where you could pull down or validate an individual's public key. This theory was initially put into practice by a group of mathematicians from the Massachusets Institute of Technology (MIT), namely Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman, more popularly known as RSA. RSA's premise was based on the understanding that when you multiply prime numbers together, there is no easy way to reduce the product back to its source. And, the larger the number, the more difficult it is to reduce, making this technique ideal for cryptographic operations that could be implemented to achieve Diffie-Helman's original and additional cryptography goals. Wow, sounds very technical. Under the hood it is quite technical mathematically but here's a more understandable explanation.

A symmetric key encryption scheme requires two or more parties to have a shared key. Think of this as a decoder ring you find in a box of cereal. As long as all the required parties have the decoder you can send encrypted messages back and forth to each other secretly. The big question about symmetric keys is how do we get the decoder ring to everyone in a way that prevents it from being compromised? Enter asymmetric key schemes that, in contrast, have two sets of keys, a private key (your secret key) and a public key (something you send about). The sender of a message uses your public key to encrypt or sign a piece of information and transmits it to you (we will get into the differences between encrypting and signing later). You use your private key to decrypt or verify the signature. Only the private key can decrypt making this a pretty good system, and quite secure.

Now that we have a basic understanding of asymmetric keys, let's talk about how this is implemented in today's technologies that you are most definitely familiar with. When you purchase an item at an online store you are normally directed to a secure page indicated by an https URL in the address bar, commonly known as an SSL protected, or secure sockets protected web page. Without your knowledge, in most cases, your browser has a very fast conversation with the server: the server presents its certificate; your browser checks this certificate against a set of accepted root signing certificates it has preloaded; your browser either accepts the certificate and starts an encrypted session or prompts you with the following message indicating it doesn't "trust" the certificate.

A quick word about "trust". With Mac OS X Server and other operating systems, you can create a self-signed certificate that you generate yourself, typically for internal use in your organization or on a test machine. This certificate in no way diminishes the encryption protection created between the browser and the server. The level of encryption is the same regardless of whether the certificate is publicly "trusted" or privately "trusted" (that is, generated by you on your Mac OS X Server). This "trust" (and I put "trust" in quotes for a reason) is created by the browser manufacturers and a group of companies that have established certain procedures and security measures that make them "trusted" by your browser's manufacturer and the public at large.

Now you see that you have been using PKI for several years and may not have known it. PKI is the technology behind the certificate: how it's generated; how it's validated; and who is or is not trusted.

TELESTREAM

Apple Design Award
Best Mac OS X Leopard Application
2008 Winner

# Record and Edit

## Your Onscreen Product Demos


Demo your software

Educate your students

Demo your iPhone app

Train your audience

## **ScreenFlow** Professional Screencasting Studio Makes It Easy

Capture the contents of your entire desktop at the same time as your video camera, microphone and computer's audio. Showcase your iPhone application with iPhone emulation capture. Sophisticated editing tools allow you to create incredible screencasts in no time. The finished result is a QuickTime movie, ready to drop right into your online product pages.

For **FREE trial** and more information, visit **www.telestream.net**

Let's take another item we are all familiar with: a credit card. I assume anyone reading this article has at least one or more cards with either of the two major card issuer's logos on it. Why is this card accepted at retailers and online stores worldwide? Why do they "trust" your card? Well, you applied for the card, the card company verified your information and then issued you a card with a unique number on it. They also have established a trust relationship with millions of vendors in both brick and mortar and online stores. This concept is quite similar to how PKI works.

In the PKI world, you apply for a certificate to an RA (registration authority), the RA validates your information and, if valid, sends a request to a CA (certificate authority) to issue you a certificate. This certificate has information about you, your organization and a serial number, just like a credit card does. You receive the certificate and use it for one of a myriad of potential uses such as securing a website, signing email, signing documents, smartcard authentication, and perhaps opening a door at your office. When you use the certificate, a VA (validation authority), aka Online Certificate Status Protocol (OCSP) responder, validates your certificate similar to the way your card is validated and checked against your available balance when you use your credit card. Just like your credit card, your PKI certificate can have a PIN (personal identification number) assigned to it to lock or unlock it. Amazingly simple conceptually, yet, as you will see, it is quite powerful and useful.

So what can we do with these neat little certificates and how can we issue our own? For starters, almost all of the services provided with Mac OS X Server can be secured using SSL, also known as TLS (transport layer security). These include iChat Server, iCal Server, Mail, OpenDirectory, VPN Server, Web Server, and Collaboration Services (Wiki/Blog/Web Calendar). They all need a certificate to function properly. Additionally, you can secure access to your wireless through the RADIUS service and a technology known as 802.1x using a certificate to ensure only your users get on the wireless network, not just anyone that figured out some shared key that is probably on a post it note somewhere in your office.

You probably weren't aware of this but Mac OS X Server automatically generates a self-signed server certificate you can use for services during its install process. This certificate can be managed from the Server Admin tool by clicking on the Certificates icon. This is the most basic of certificate administration tools. There are several ways you can issue and manage certificates. For smaller environments, Apple provides the certificate assistant located in your `/System/Library/Core Services` folder. In next month's article, we will delve into setting up your own certificate authority and issuing certificates using this tool. Also, for larger installations, there is an open source project called EJBCA (Enterprise Java Beans Certificate Authority) that offers both free community support and paid for corporate support and training. To download and install EJBCA go to www.ejbca.org. Support, training, and customization are provided by PrimeKey Solutions (www.primekey.se). EJBCA

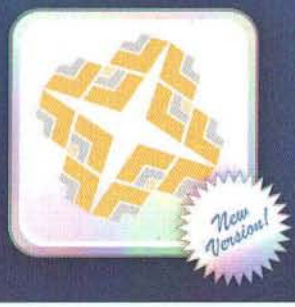
JAMF software

# One Company. { Four

## Enterprise Solutions

for Education, Government
and Commercial Institutions.






### Casper Suite

A complete, best of breed client management platform that automates the most common IT functions.

### Imaging Suite

The new imaging standard for the management and deployment of images.

### Recon Suite

Web based, cross platform inventory solution.

### Composer

Simple snapshot and drag & drop package creation.

**Now, more than ever, it's time to get serious about your Macs.**

No matter what your starting point is, we have options that will simplify the management of your Macs.

At JAMF Software, we make the tools you need to manage your growing Mac network. No other company offers such a comprehensive toolset with the expertise to back it up.

Contact us today to learn more.
**www.ManageMyMacs.com**

★ Made & Supported in the USA

will be described in detail in a future article. For now, just take a look at your Mac OS X Server and play around with the Certificate function to create some self-signed certificates and use them to test some services. Be careful not to delete the default certificate if it is already in use to prevent disrupting anyone's ability to connect to a given service.

## Conclusion

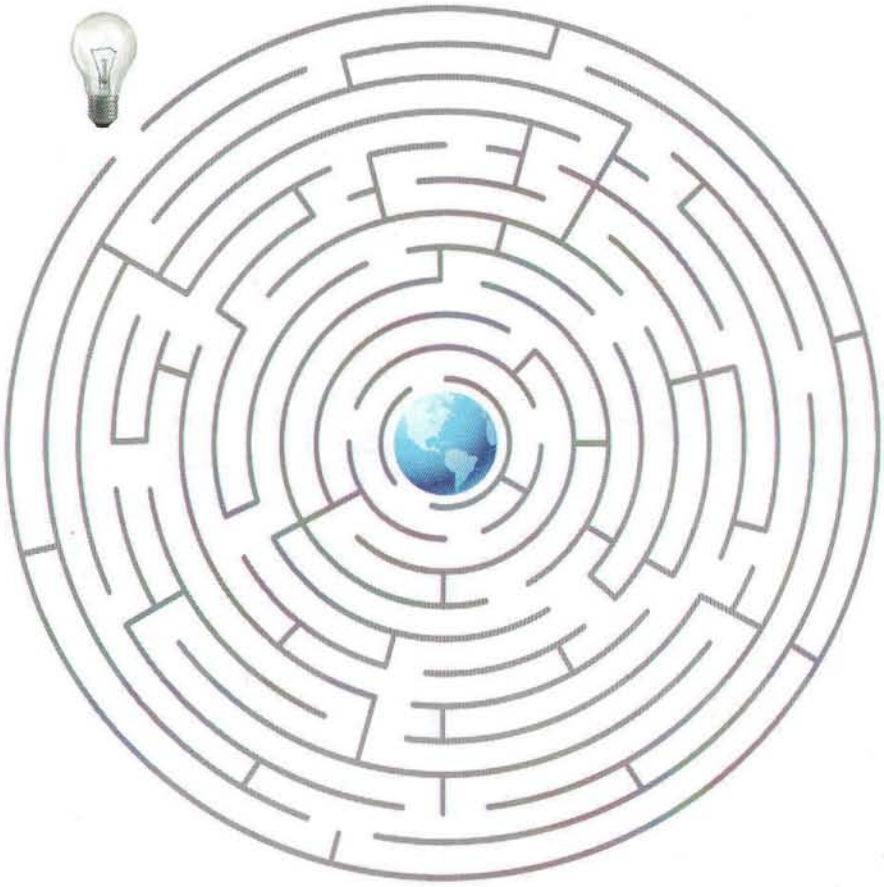
So we have started down the wonderful road to public key infrastructure (PKI). With this basic understanding under our belt, we can build our own certificate authorities, generate our own web and other certificates and learn how to use PKI for some pretty neat security functions like email and document signing. Till next month, stay secure and happy computing.

## About The Author

***Michele (Mike) Hjörleifsson has been programming Apple computers since the Apple ][+, and implementing network and remote access security technologies since the early '90s. He has worked with the nation's largest corporations and government institutions. Mike is currently a certified Apple trainer and independent consultant. Feel free to contact him at mhjorleifsson@me.com***

# Access a worldwide market

## We'll lead your product through the maze



## Reach a global retail audience with Smith Micro Software.

Working on the next great Mac Application? Love to see your App on store shelves and catalog pages but you don't know where to start? Get into stores with Smith Micro. We have over 20 years experience publishing and distributing award-winning products for Mac and Windows markets. We do it all—package design, marketing and manufacturing, so you don't have to.

Great products published by Smith Micro:


smithmicro
software

Talk to us to get started:
email **bizdev@smithmicro.com**

# Sharepoint 101

## "This ain't your mother's file server," or, "What administrators need to know about Microsoft's webified tool for working together"

*by William Smith*

## Moving away from the shoebox mentality

### Introduction

What do users in both small and large organizations need to share today? Memos, mail messages, pictures, video clips, sound bytes, documents, original files, derivative files, receipts, spreadsheets, presentations, diagrams and more and more *stuff*.

Small companies and large enterprises have progressed well beyond pieces of paper, manila file folders and inter-office vacuum tubes to storing data on centralized servers. But even with centralization, a large server with a terabyte or more of storage space is essentially just a gigantic shoebox. Folders are shared and filled with more folders and sub-folders to organize everything. Documents and other electronic files are placed within those folders and sub-folders. File servers start as nice, neat, organized shoeboxes of *stuff* but over time those piles get disorganized and scattered or the servers themselves get overfilled, requiring the purchase of yet more servers.

Microsoft's solution to the shoebox problem is SharePoint Server, a virtual California Closets for file servers. SharePoint is a *content management system* or, "CMS." A CMS's job is to provide organizational structure within a server. This structure comes in the form of separating document libraries from video libraries or it comes in the form of separating financial data from creative data. A good CMS will let you do both, letting you see the same information presented in a variety of ways.

SharePoint is a web server presented to the end-user through any modern browser such as Firefox, Safari or Internet Explorer for Windows. It is designed to allow, even *encourage*, everyone to contribute content to the site without having to know web design. It is designed to keep users within the boundaries of their shoebox areas too.

### What can I do with SharePoint?

A great use for SharePoint is maintaining a company-wide Intranet site. Departments are as varied as they are plentiful and each will have unique needs. Assume that web browsers throughout the company all default to an intranet webpage displaying company news, an address book and a calendar of events. That webpage could then serve as a portal for each department such as Human Resources (HR), Information Technology (IT), Creative Services and Facilities, just to name a few.

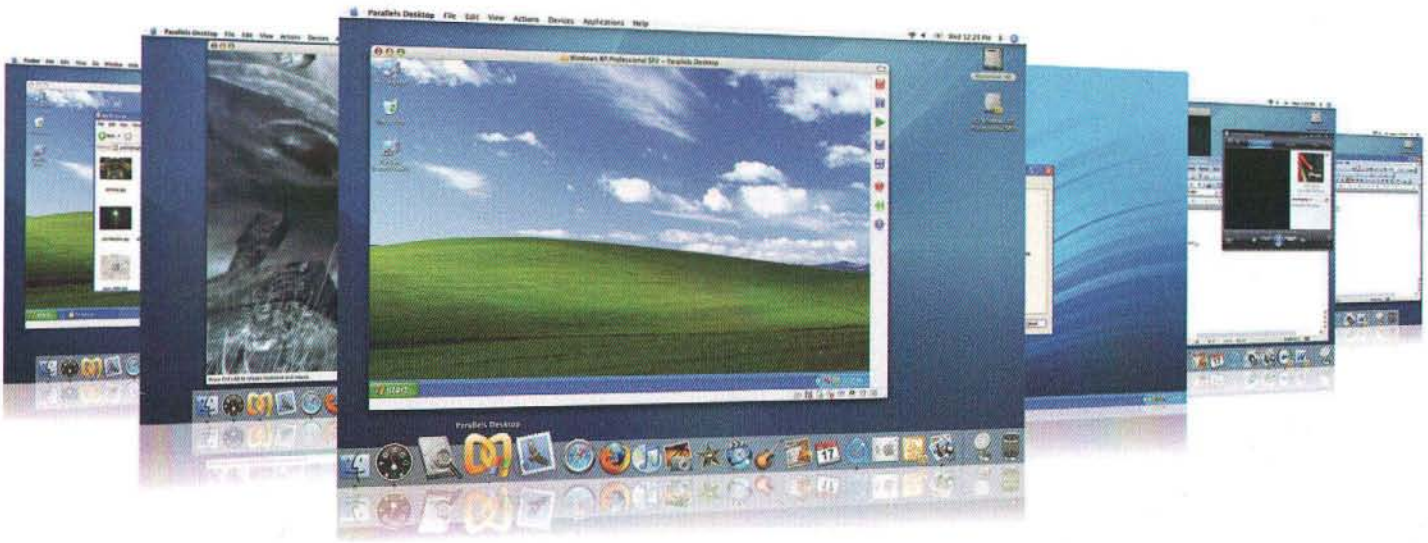
HR can provide its employees with access to employee forms, medical leave balances and other information through its own SharePoint site, which is a sub-site of the overall Intranet website. Most of this content is static and doesn't change often but when it does have to change, one person within HR can update the site herself without having to know any web coding. She can edit a page within the SharePoint webpage or choose to download and edit Word documents that have been posted. When the changes are complete, she simply saves her changes or uploads the edited documents where they are immediately available to everyone.

IT has a need to control its requests for new hardware and software. Rather than having users call the Help Desk to make these requests, the assets management group can post forms on IT's own sub-site of the Intranet site. The forms can include required fields such as cost center, approving manager, computer platform, etc. These forms will not only reduce calls to the Help Desk but will ensure that all necessary information is included in the request prior to submission. Submitting the form can trigger an automatic E-mail message to the IT assets management group, which can review the requests and place the orders.

Creative Services often receives requests from HR, Marketing, Sales and other groups for company logos. Some groups need the logos in color while other groups need black & white and some groups need logos in EPS format for print while other groups need JPEGs for websites. The Creative Director can include these image files on its SharePoint site, again a sub-site of the Intranet website, along with directions for use and a stern warning against stretching and distorting. If a Sales user needs a color TIFF file with a Windows preview for a PowerPoint presentation but one doesn't exist, then he can make the request by completing an online form.

# Plays well with others.

## Run Windows and Mac OS X at the same time - without rebooting.




**With Parallels® Desktop 3.0 for Mac, you can:**

- Automatically open Windows files with Mac software and Mac files with Windows software
- Bulletproof your virtual machine from Windows system crashes and malware
- Run selected Windows-only 3D games and applications with 3D graphics support
- Browse through Windows files and folders without launching Windows
- Move your existing Windows PC to a Mac without losing data or reinstalling any software
- Completely share files and folders between Windows and Mac

Parallels™

Parallels® Desktop 3.0 for Mac provides even more ground breaking features and capabilities than previous versions. Already The #1 Selling Mac System Utility according to NPD Techworld (09/06 – 12/07) and recently named an InfoWorld Magazine "2008 Technology of the Year", Parallels® Desktop is trusted by over 800,000 users world wide.

Want better Windows and Mac integration? Want to run the hottest PC games and graphics software? Worried about security and system mishaps? Parallels® Desktop 3.0 for Mac includes 50+ new features and enhancements, including a number of integration features and virtual machine utilities unavailable anywhere else.

To discover why Parallels® Desktop is the leading desktop virtualization solution for running Windows on a Mac, visit us online or call us at 1 (425) 282-6405.

PC WORLD 100 BEST PRODUCTS of 2007 — Macworld BEST OF SHOW 2007

Download a trial today at www.parallels.com/download

Website – www.parallels.com Email – sales@parallels.com Phone – 1 (425) 282-6405

Facilities is in charge of this year's company picnic and recalls that employees didn't care for last year's choice of location. Planning well in advance, they decide to post a survey listing three choices that fall within budget. This year the employees get to choose whether they'd like an outdoor picnic at a local park, an indoor buffet at a local game venue or a softball tournament at an outdoor recreation complex. Hyperlinks to each venue's external website are available to assist in decision-making. The results of the survey are immediately and automatically available to the Facilities group at any time during the survey or when the survey is complete.

SharePoint can even be used as a customer-facing external website. Because SharePoint is delivered to the end-user via a web-browser, nothing dictates that it cannot be used outside of a company network. With some customization using the free SharePoint Designer tool, a SharePoint site doesn't have to look like a default SharePoint site.

## Enhanced file sharing

SharePoint is not a complete replacement for file sharing servers but it is ideal for protecting the integrity of files.

Assume the Finance department produces a spreadsheet with quarterly results and that several department heads must approve. The Finance Director had a difficult time finding a secure location on the company file server that all necessary users could access but that restricted access for everyone else. Eventually, he resorted to using E-mail, which was not only insecure but allowed each department head to make changes directly in the spreadsheet, resulting in multiple versions.

Using SharePoint, the Finance Director can upload the file to a secure website protected using Secure Sockets Layer (SSL). He can then assign reviewer permissions to the document for only those who need to see it. No one else in the company has access. Furthermore, those reviewing the document can add comments but cannot change the content. Once reviewed and approved, he can keep the document available online for historical purposes.

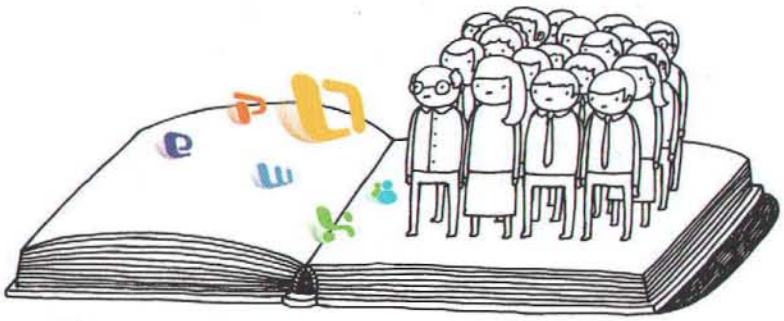
## Replacing Exchange public folders

According to Microsoft, SharePoint is the replacement for public folders, a feature that was de-emphasized in Exchange Server 2007 and is losing ground going forward in Exchange Server 2010. Public folders enable users to share mail, calendars, contacts, tasks, notes and documents with each other through Outlook, Microsoft's E-mail application for Windows. Entourage, however, Microsoft's E-mail client for Macintosh is limited to just mail, calendars and contacts.

The likelihood of Entourage ever supporting more with public folders than it does now is practically nil. Furthermore, by moving these items from a mail server to a SharePoint server, mail administrators will not be tasked with supporting file sharing within E-mail.

## SharePoint for your company or just for you

Mac users who work in a Windows world may already have SharePoint services available within their company or at least have



Everyone on the same page

Work together. Different machines? Different platforms? No matter. You can all speak the same language.

SimplifyYourWork2008.com

Microsoft® Office:mac 2008

the infrastructure necessary to implement SharePoint at an enterprise level. However, smaller organizations and even individuals with limited resources have options for getting SharePoint too and for very reasonable prices.

For individuals and small organizations without a server budget, online service providers offer SharePoint services over the Internet. This is known as "SharePoint hosting". Pricing starts as low as $10.00 per month for up to 500MB of storage space with unlimited users and includes 24/7 support. Plenty of hosting services offer 30-day free trials for you to evaluate SharePoint for your own needs.

For those larger organizations where IT has been brought in-house and is supported either full-time or part-time, then Microsoft Small Business Server (SBS) may fill their needs. SBS is an all-in-one Windows Server product that offers not only file and printer sharing but SharePoint, remote connectivity and Exchange for E-mail. Full SharePoint services are provided with SBS, however, enterprise-level options like SharePoint farm servers aren't available. This product really is intended for "small business".

SharePoint Server can be virtualized. Mac shops that want the feature set of SharePoint but want to utilize their existing XServe infrastructure may want to consider running SharePoint on bare-metal virtualization.. A newer Intel XServe with beefy amounts of RAM and processing power can run multiple Mac OS X and Windows virtual machines. Microsoft offers a 32-bit trial version and a 64-bit trial version directly from their website for testing.

Microsoft itself offers Office Live Workspace online, which is very similar to SharePoint in the way it functions. It is currently in beta but open to the public for free with up to 5GB of space for storage. Office Live Workspace can be used for both work and home and it and offers many templates for creating a shared space quickly and easily. It is available at http://workspace.officelive.com.

## Moving into a SharePoint site

### Out of the box

Once SharePoint has been installed, the administrator must set up the first site. Usually, this is a site based on one of four groups of templates, which come pre-installed for various team sites. Figure 1 shows the default Team template from the Collaboration set of templates. It comes pre-populated with placeholder information.

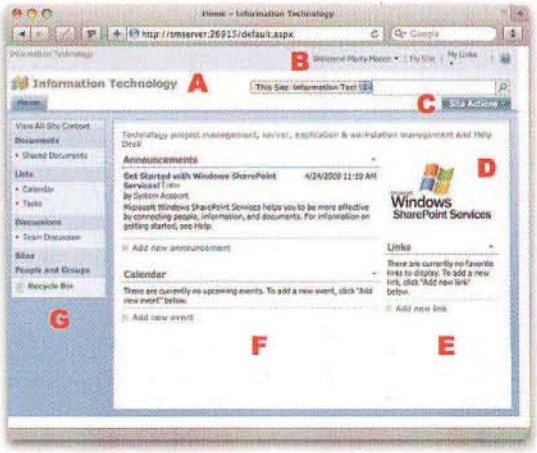
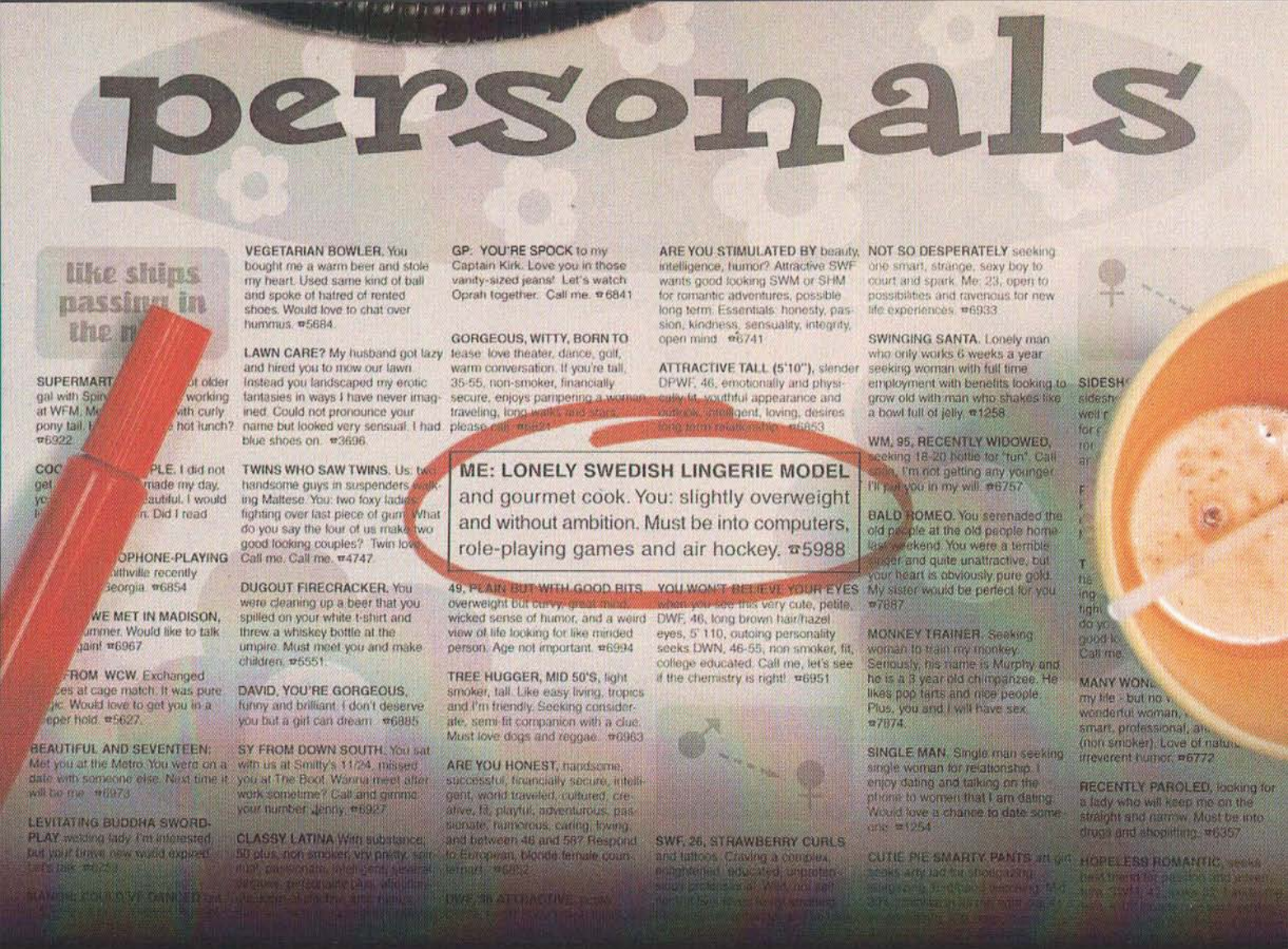



**Figure 1. Default Team template from the Collaboration set of templates**



The default webpage is divided into several sections:

A) Top link bar – Includes the name given to the site during setup, user links, navigation tabs and search.

B) Login and personalization – Allows users to change accounts, log out and customize the appearance of the site for their own needs.

C) Site Actions menu – The site owner will see this menu, allowing him to customize the appearance of the site as well as the structure of the content. He can make this menu unavailable for general visitors.

D) Site Image – The site owner can apply a custom graphic for personalization.

E) Right Zone – The right column or "zone" of the site.

F) Left Zone – The left zone of the site.

G) Quick Launch bar – A navigation bar for browsing the site.

## Site Actions menu

The Site Actions menu is probably the most important menu for a site owner. It is used to modify the entire site from simple name changes to moderate page layout to complex navigational links. Only the site owner and anyone he allows will see this menu. Otherwise, anyone visiting the site will see it as the site owner has chosen to present it.


**Figure 2. Site Actions menu**

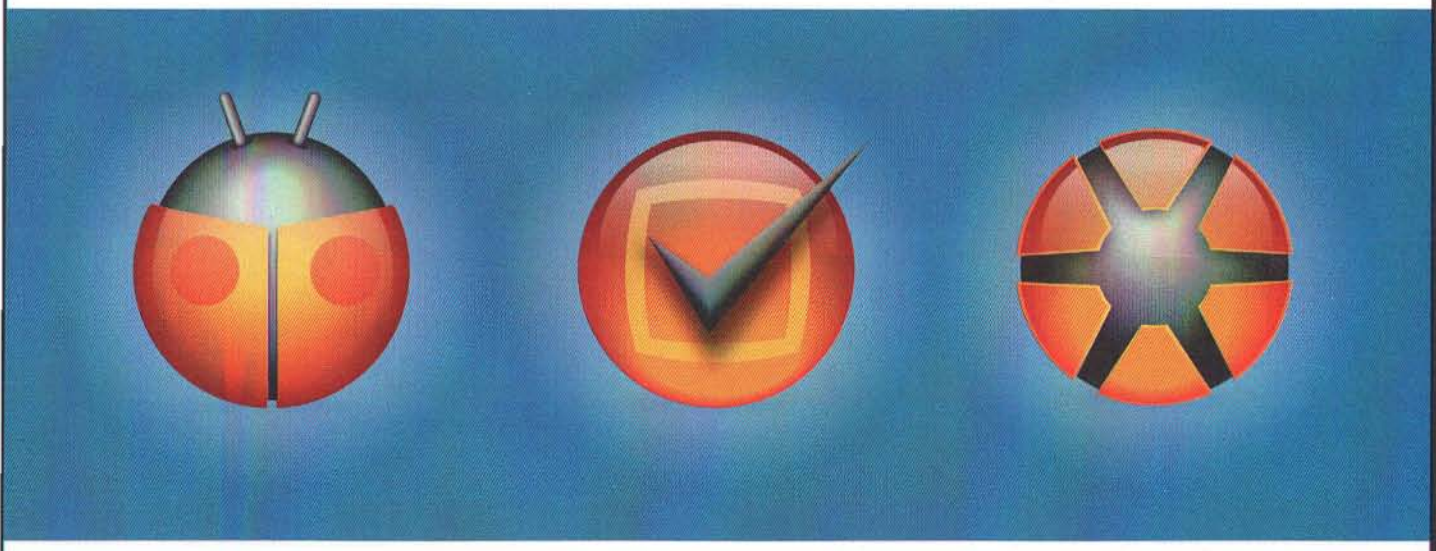
From the Site Actions menu the site owner can select the Site Settings command to control read/write access to the site. He can also adjust the look and feel of the site by changing the Title, site theme and Quick Launch navigation bar and he can even save his new look as a template. The Site Settings page is where the site owner can add Web Parts (more on those in a minute) to the current page as well as apply regional settings such as the current time zone, date & time appearance and workweek days and hours. The site owner can even delete the site itself from this page.

## Parts is parts

What makes SharePoint easy for end-users? How can one department customize SharePoint for its needs while another department is customizing SharePoint for its completely different needs? The answer is *modularity* or what SharePoint calls *Web Parts*.

SharePoint includes more than 30 Web Parts that can be arranged on a page in hundreds of ways. Web Parts include announcements, calendars, links, tasks, team discussions and

Download free trials of TestTrack Pro, TestTrack TCM, and Surround SCM at:
**www.seapine.com/mactech**





# Seapine ALM solutions for serious Mac OS X development

**Stay on track with Seapine's Mac OS X-native development tools.** Designed for the most demanding software development environments, Seapine's Mac OS X-native application lifecycle management (ALM) solutions are scalable, feature rich, team-based tools that can be used separately for superior issue tracking, test case management, and software configuration management–or seamlessly integrated for more efficient control of your software development process.

### TestTrack Pro
Issue & Defect Management

- Track defects, change requests, feature requests, and other project-related issues.
- Tailor workflows, including events, states, and transitions, to your development process.
- Stay informed and on track with flexible reports, email notifications, and escalation rules.
- Create and link defects with failed test runs in TestTrack TCM for better traceability.
- Link defects and change requests to source code changes in Surround SCM and other SCM tools.

### TestTrack TCM
Test Case Planning & Tracking

- Manage thousands of test cases, select sets of tests to run against builds, and process the pass/fail results using your development workflow.
- Ensure all steps are executed, and in the same order, for more consistent testing.
- Know instantly which test cases have been executed, what your coverage is and how much testing remains.
- Track test case execution times to more accurately estimate the time required to test applications.

### Surround SCM
Configuration Management

- Control who changes source files, and track what has changed and when.
- Leverage file-level workflow to track and manage the state of individual files.
- Facilitate parallel development with advanced virtual branching, private branches, and workflows.
- Notify team members of new files, assignments, and changes by email.
- Quickly access the latest files with shadow folders, hyperlinks, and Finder integration.

Seapine Software™

**QA Wizard® Pro** Automated Testing

**Seapine CM®** Change Management

**Surround SCM®** Configuration Management

**TestTrack® Studio** Test Planning & Tracking

**TestTrack® TCM** Test Case Management

**TestTrack® Pro** Issue Management

shared documents to name a few. Each Web Part is a mini application dedicated to just one function on a SharePoint site. By arranging various Web Parts on a page, the site owner can create a custom portal that can be as complex or as basic as he chooses. Web Parts can be placed in the left zone or the right zone of the page, can be resized and include as much or as little detail as needed. This is reminiscent of well-known Internet portal websites such as iGoogle or Windows Live.

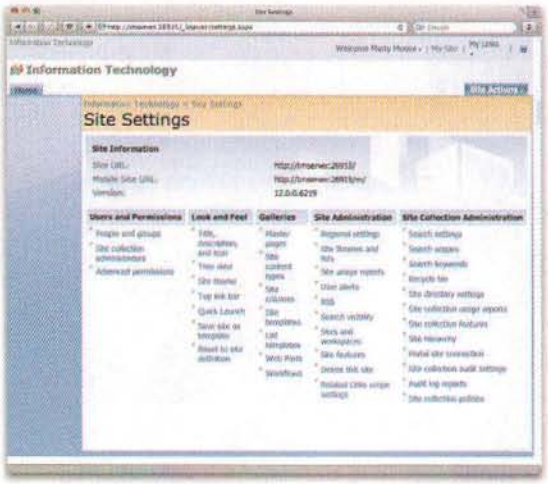



**Figure 3. Site Settings**

## Customizing Web Parts

The Site Actions menu also leads to the Edit Page command, which the site owner or anyone he allows can use to modify the layout or Web Part content of each page within the site. While in SharePoint's Edit Mode, the editable portions of the page are highlighted and the Left and Right sections of the page display Add a Web Part buttons.

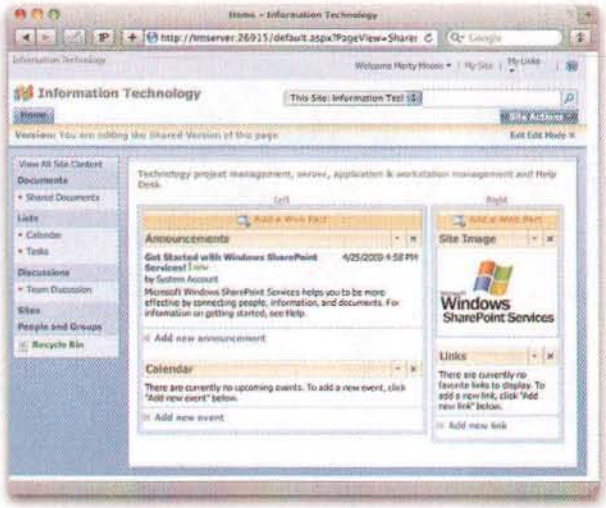



**Figure 4. Edit Mode**


# MORE PACKED!

**The MacTech DVD - Volumes 1.01-24.12** is packed with more than ever before -- almost 3100 articles from more than 280 issues **(1984 - 2008)** written by over 850 experts, all 29 issues of Apple's develop, 21 issues of FrameWorks magazine, 100+ MB of source code, MacTech Viewer, working applications, full documentation, demos for techs, ***and more!*** The latest version includes all kinds of third party applications, videos, and demos.

Everything is displayed in the very fast, very searchable ***MacTech Viewer!*** An application that has been designed specifically with Techs in mind. Search quickly through almost 25 years of of great information provided by MacTech. Information to save you time, and make your life easier.


Mac Universal

Requires Mac OS X v. 10.4.5 or later

Toll Free 877-MACTECH, Outside US/Canada: 805-494-9797 • http://www.mactech.com/dvd/

*(A quick note about web browsers: While the same SharePoint site displayed in Internet Explorer for Windows and Safari for Mac should look similar, they will not always be the same. IE for Windows is considered a Level 1 browser, which means it can take advantage of all the browser features supported by SharePoint. In particular, IE for Windows can take advantage of ActiveX controls allowing the user to view more information and even drag and drop Web Parts during arrangement. Firefox and Safari are considered Level 2 browsers. All of the functionality is present but just not accessible in the same way. Other browsers may work but are not supported.)*

To the right of the Title of each Web Part are two buttons. The first button is the Edit button. Clicking this button displays the properties for that particular Web Part in a temporary pane to the right of the web page. Clicking the Edit button for the Site Image Web Part, for example, displays the Site Image properties such as the path to the image file, ALT text, alignment, appearance, layout and more.

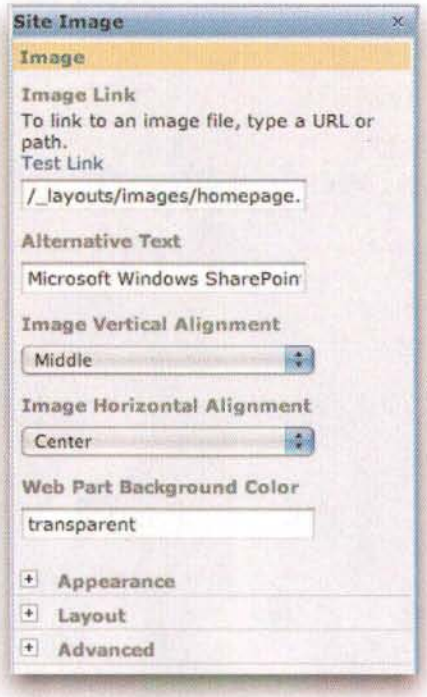



**Figure 5. Site Image properties**

For the most part, the attributes of every Web Part are the same. These attributes define the Zone or location of the Web Part on the page (Left or Right), its order within the Zone or Zone Index, the Web Part's Title, who can see the particular Web Part, etc.

The second button next to the Title of each Web Part is the Close button, which removes the Web Part from the page.

## Templates

Deciding which Web Parts to use and how to arrange them can be a daunting task, so Microsoft has pre-populated SharePoint with several default templates, arranged with various Web Parts, and they offer more free templates for download from their website. The default templates are grouped into four categories:

- Collaboration
- Meetings
- Enterprise
- Publishing

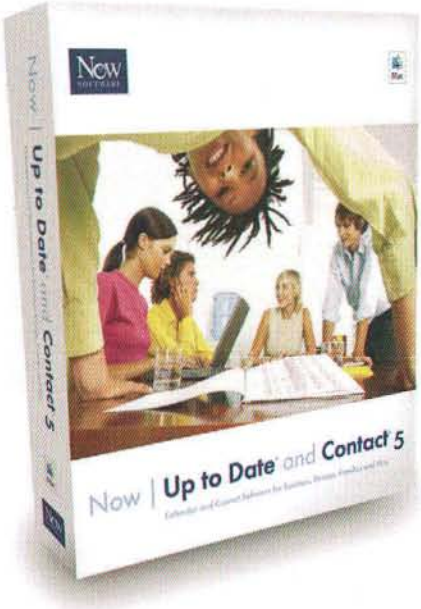
The Collaboration set of templates includes a Team website, which is ideal for a group of people needing to share documents and information. They also include templates for a Wiki, Blog and for Documents.

The Meetings set of templates includes a basic template for managing agendas, attendees and documents as well as more specialized templates such as a Decision Meeting template for recording decisions and creating tasks.

# Now **scheduling** and **contact management** for your entire organization.

# Now | **Up to Date**® and **Contact**® **5**

**Calendar and Contact Software for Business, Groups, Families and You.**




Is this project on schedule? When are you available to meet about the systems upgrade? Where are all the field techs today? When was the last time anyone talked to our biggest customer?

Virtually all groups live (or die) by their abilities to meet deadlines and keep track of their customers, prospects, and vendors. Few small companies or even departments of big companies have the tools they need.

Now Up-to-Date & Contact might just be the calendar and contact software for you. It's time-tested and used by more Mac-based companies than any other solution. And it's cross-platform–available for your PC users, too. It's easy to install and manage and simple for your employees to understand and use.

Using Now Up-to-Date & Contact you can schedule meetings for multiple users, view multiple, simultaneous calendars, and reserve rooms and resources. You can share contact information about your customers, prospects and vendors. And using our free server software you can set it up in minutes and share with users in the office or from anywhere with an internet connection.

**Call us now at 866-527-0556 or email us at mactech@nowsoftware.com and we'll send you our free evaluation kit, including the book that will make it all easy, "Take Control of Now Up-to-Date & Contact" from Take Control books!**

NOW SOFTWARE

**Phone:** 866-527-0556

**Web:** www.nowsoftware.com

The Enterprise templates are for larger scale sites such as a Documents Center and a Records Center. A Site Directory template is available to set up the Intranet home page mentioned earlier. This is ideal for listing other sites in your organization and even includes a top sites feature as well as a site map.

Finally, the Publishing templates include an Intranet portal for internal site management, including search features, as well as an external or Internet-facing set of pages. External sites are expected to have many readers on the outside of the organization with content producers residing on the inside of the organization.

Blank site templates are included as well for those who want to start their SharePoint sites from scratch.

## Putting away the dishes

### Content

So far, everything has been about the layout, look and feel of the SharePoint site. What about content? SharePoint can hold most any type of file or data but how can it help its visitors find those items?

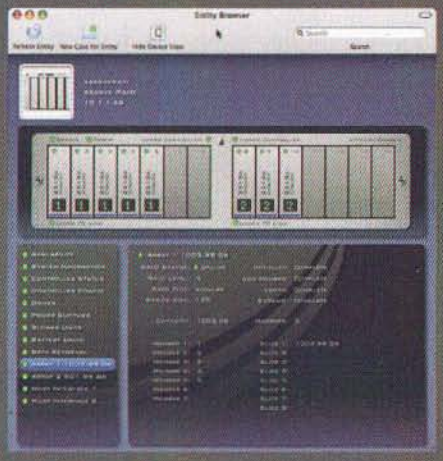
On the Home page just below the Home button is the Quick Launch bar. This is SharePoint's customizable navigation tool and the door to putting stuff into it. The very first link is View All Site Content. This link leads to a sort of site map and displays all the Document Libraries, Picture Libraries, Lists, Discussions, Surveys and other Sites within this site.

With an overview of everything in the site, this is the perfect place to create new *Libraries*, which are simply collections of files. SharePoint offers Libraries for different types of files. A Document Library is suited for files that must be downloaded and viewed in the appropriate application. This includes Microsoft Office documents such Word and Excel files. A Picture Library is suited for displaying information such as picture size, file size and even a preview for JPEG, TIFF and other graphic files. A Slide Library is specific to presentation type files such as PowerPoint or Keynote.


**Figure 6. All Site Content**

# Network, Server and Appliance Monitoring.

# For Mac OS X



Xserve RAID

Xserve (Intel & G5)

Airport

## Lithium Network Monitoring Platform

Lithium can now monitor your Xserve, Xserve RAID, Qlogic switches, Airports, Mac OS X Server... and everything else in your network.

www.lithiumcorp.com

lithium Corp.

While the default settings are just fine for most uses, each Library can be extensively customized to share even more information about its contents. For example, assume the Shared Documents Library contains a Word document with a filename "Fiscal Year 2009 Summary", which appears in a list of 20 documents. It also has an author, a creation date and a file type (Microsoft Word file) that each appear in separate columns. By default, these are the only columns of information the site visitor will see.

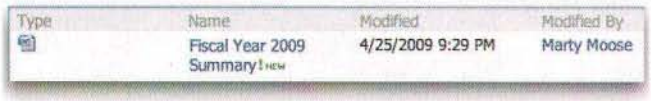



**Figure 7. Default Document Library columns**

By selecting the Create Column command from the Settings menu, the site owner can add columns to the list.

Assume the site owner has added a Status field with three options (*New*, *Review* and *Complete*) that can be selected when the document is uploaded to the site or after it is already uploaded. This will allow the site visitor to sort the columns by Status to bubble-up all the documents marked for "Review".

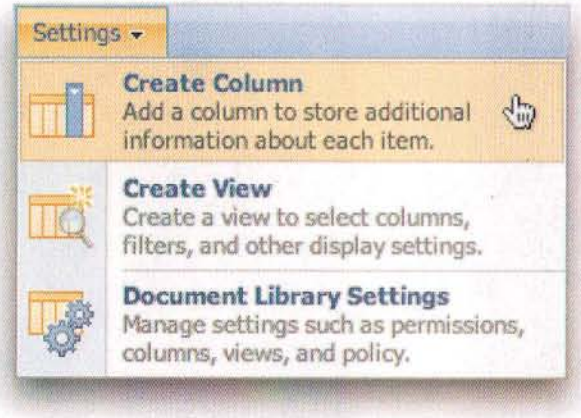



**Figure 8. Document Library Settings**

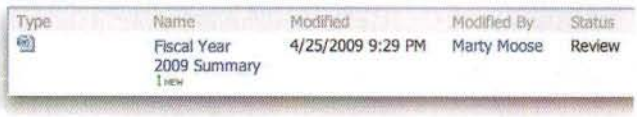



**Figure 9. Modified Document Library columns**

Columns can display data from a variety of input options such as single-line fields of text, multi-line fields of text, number fields, drop-down menus, checkboxes, radio buttons and even calculations based on other fields or pieces of information. When a contributor uploads his file he will be presented with a page to enter all the metadata about the file.

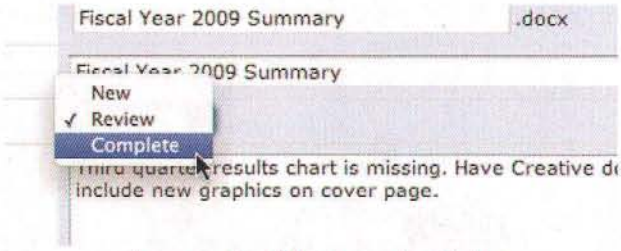



**Figure 10. Modified metadata fields**

Note that not all information that's included about a file has to necessarily be visible in a column. If the name of the file and its Title are essentially the same then the site owner may choose to simply not display the Title column. But what if information is valuable to some visitors but not to others? Under the Settings menu is also a Create View command (see Figure 9). The site owner or anyone he allows can select this command to display alternate views of this information.

Assume that the Creative Services group maintains a Picture Library for other departments to use. The Web Services department requires 72 DPI JPEG images whereas the Print Services department requires 300 DPI TIFF images. Rather than create and maintain two separate Libraries, the Creative Services group can simply create one view called "Web" and another view called "Print". Visitors to the site can select the view of their choice (see Figure 12).

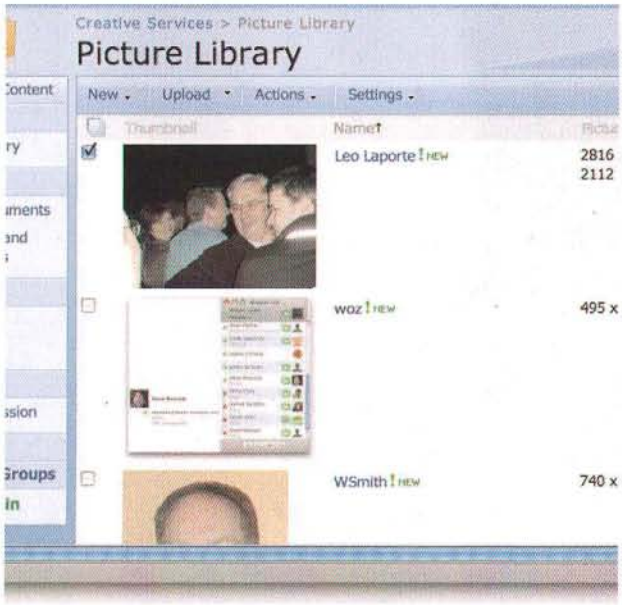



**Figure 11. Custom Picture Library**

## Even more content

With libraries of documents, photos and slides with a wiki and a couple of blogs thrown in for good measure, how can so much information keep from becoming another shoebox of stuff? The answer may be more sites!

Not only are web pages modular but so too are sites. SharePoint is an excellent choice for creating an Intranet because it

Free Edition Available

# Help Has Arrived.

# Web Help Desk™

## Powerful Software for Service Management

- ✓ On-Demand or On-Premise Plans
- ✓ Incident & Problem Management
- ✓ E-Mail to Ticket Conversion
- ✓ Asset & Inventory Management
- ✓ Knowledge-base & Self-Help
- ✓ LDAP / AD Synchronization
- ✓ Customer Service Web Portal
- ✓ Approval & Change Workflow
- ✓ Performance Reporting Features
- ✓ Customer Satisfaction Surveys

Mac Universal

MySQL ORACLE Microsoft SQL Server

**Save 10% · Special Code: MTM0109 · Contact sales for details**

www.webhelpdesk.com • 1-877-943-0008 • sales@webhelpdesk.com

can contain sites within sites within sites. Responsibility for maintaining each site can be delegated to members of each group so that content can be published as quickly as possible and maintained by many users rather than just a few.

Assume the IT department is made of several smaller groups: Application Management, Development, Help Desk, Project Management, Server Management and Workstation Management. Each has a need to publish and maintain its own documentation, some private and some public. For example, the Help Desk has a need to publish forms for hardware and software requests while the Project Management group has a need to maintain tasks and timelines.

By revisiting the View All Site Content link again in the Quick Launch bar, the site owner can click the Create button and instead of creating a new Library or List, he can create a new sub-site. He can create one for each group within IT and their sites will now appear as tabs in the link bar at the top of the page (see Figure 13). The Home button to the far left will always return visitors to the top-level site.


**Figure 12. Site tabs in top link bar**

## Making SharePoint easier to use

### Office for Mac and the DCC

SharePoint is unabashedly a Microsoft product and Microsoft makes integrating most of its server technologies with its workstation technologies very easy. Office for Windows can seamlessly work with SharePoint and Internet Explorer to give its users an uninterrupted experience when editing documents. When clicked, a link to a Word document will download a copy of the document and Word for Windows will display its contents in seconds as if the user had opened the file from a local file server. Editing and saving an Office document will quickly upload changes to the server without a blip. The same applies to Excel and PowerPoint files.

Macintosh users don't get such an integrated experience. Even with Safari's preference to Open "safe" files after downloading enabled, clicking a link will at most just download the Microsoft Office document to the user's Downloads folder. The Mac user must then locate the file and double-click it to view or edit. When finished editing, he must return to the SharePoint site to upload the document, saving over the old one if necessary.

At the Macworld Expo 2009 in January, the Macintosh Business Unit (MacBU) at Microsoft unveiled a new product called the Document Collaboration Companion (DCC). The DCC promises to smooth out some of the wrinkles that Office for Macintosh users face when dealing with SharePoint sites and

# LIBERATE YOUR LAPTOP

A SINGLE SOURCE SOLUTION PROVIDER
FOR ALL YOUR TECHNICAL REQUIREMENTS

**PARTS + UPGRADES + SERVICE**

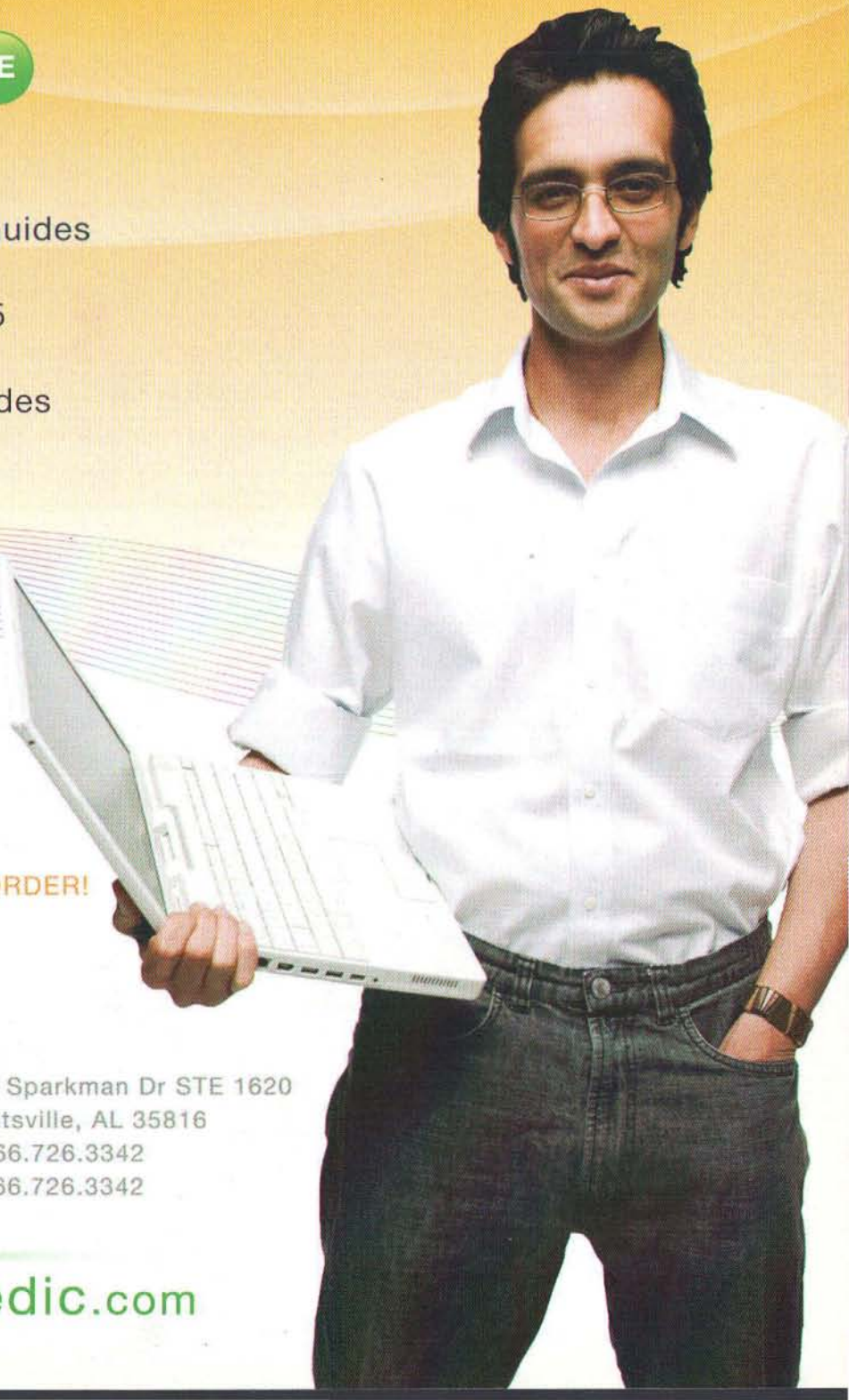
## INDIVIDUALS

- Free Online Do-It-Yourself Guides
- Send-In Repair Service
- Power Adapters from $24.95
- Batteries from $94.95
- Replacement Parts & Upgrades
- Diagnostics
- Hardware Installation

## DEALERS

- Reseller Accounts
- Outsource Repairs, Increase Profits
- Blind Drop Shipments

USE COUPON CODE MACTECH5
TO RECEIVE **5% OFF** YOUR FIRST ORDER!



PowerbookMedic.com
*Your Mac ~ Our Patient*

555 Sparkman Dr STE 1620
Huntsville, AL 35816
**T** 866.726.3342
**F** 866.726.3342

www.PowerbookMedic.com

even goes so far as to enable working with Office Live Workspaces. According to Microsoft's Mactopia website, the DCC is currently in private beta and will be released later in 2009.

### Putting the lid on the shoebox

Little or nothing has been said so far about advanced design, security, searching, personal pages, team discussions, project management, group calendaring or many of the other features of SharePoint. While individuals may never use half or even a quarter of its feature set, a small company can easily take advantage of many of the tools it provides. An enterprise can make use of most, if not all, of its features.

Half the battle of data management is filing content so that information doesn't get lost or become difficult to find. As part of a larger set of collaboration tools—Microsoft Office for documents, Exchange Server for E-mail and Office Communications Server for instant messaging—SharePoint is poised to make sharing and organizing information within a team environment or a company network quick and efficient. It really is a product that can shine when individuals are allowed to play in a sandbox, so administrators should be encouraged to create sandbox sites for their more creative power users. They will discover more when they are able to use as much of it as possible and be encouraged to continue using it.

## More information

California Closets
http://www.californiaclosets.com

SharePoint Server homepage
http://office.microsoft.com/en-us/sharepointserver/default.aspx
SharePoint Designer
http://ww.microsoft.com/spd
Office Live Workspace
http://workspace.officelive.com
iGoogle
http://www.igoogle.com
Windows Live
http://home.live.com
Free SharePoint Templates
http://technet.microsoft.com/en-us/windowsserver/sharepoint/bb407286.aspx
Document Collaboration Companion Beta
http://www.microsoft.com/mac/itpros/dcc.mspx

## About The Author

***William Smith is a technical analyst supporting Macs in a Windows world. He works in the Twin Cities and enjoys educating folks that Macs and Windows really can get along. He's a seven-year Microsoft MVP, co-founder of the Entourage Help Blog http://blog.entourage.mvps.org and enjoys shopping at The Container Store. You can reach him at bill@talkingmoose.net.***

SecuTech
SecuTech Solution Inc.

# Unikey Time

## A Real Time Clock Inside


## Easy Solution

On-key clock counts date and time

Driverless technology reduces customers 'support work

Both automatic (envelope without source modification) and APIs protection

Numerous sample codes in various programming languages

## Greater Flexibility

Protect software, flash (swf and flv) and video files

Protection in local computers or over a network

Support all popular operating systems, including Windows, Linux, MacOS and Free BSD

Remote update and real time functionality

OEM enables flexibility of case, label, and color

## A Cost-effective Choice

Competitive pricing

Experienced and efficient technical support

Life time warranty



• NORTH AMERICA: 1- 888-259-5825

• BRAZIL · EGYPT · FRANCE · GERMANY · INDIA · ITALY · JAPAN · MIDDLE EAST · SERBIA · TURKEY

SecuTech Solution Inc. | www.eSecuTech.com | info@eSecuTech.com

by Dave Dribin

# Chips or Fries?

## Handling User Preferences

## Introduction

Last month, we covered how to display windows and sheets using canned alerts via the `NSAlert` class as well as custom windows and sheets stored in separate nibs and displayed with `NSWindowController` subclasses. This month, we're going to cover how to handle user preferences, as well as how to implement a preferences window that works like most Apple-supplied applications.

I'm going to show you the end result, and then, we'll start filling in the code. The application contains a window with a simple custom `NSView` subclass that displays your favorite word, as shown in Figure 1:

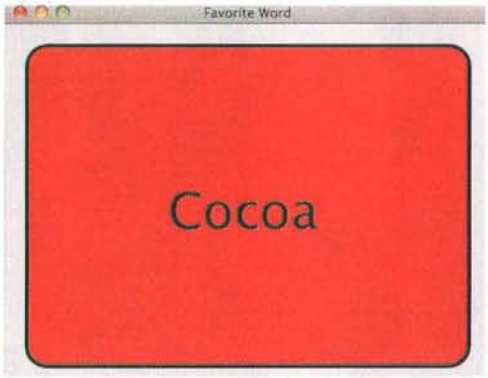



**Figure 1: Favorite Word window**

We previously covered custom views, so there's not a lot new, so far. However, I'd like to add in a preferences window so that the user can change their favorite word, as shown in Figure 2:

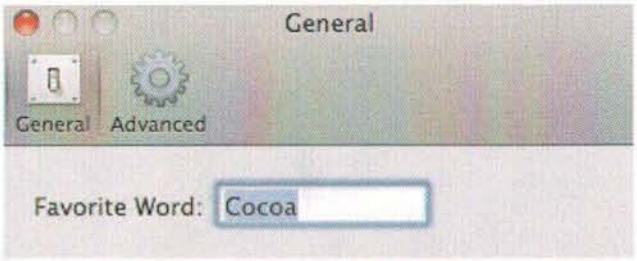



**Figure 2: General preferences**

But, let's not stop there. We should let the user customize the background color and text alignment, too, as shown in Figure 3:


**Figure 3: Advanced preferences**

As you can see, the preferences window is separated into two panes: the General pane and the Advanced pane. It is fairly typical to setup multiple panes in preferences windows to separate options into groups. However, even though this is the standard practice for preferences windows, Apple does not provide us with a ready-made preferences window. We'll have to write a fair amount of code to emulate these standard windows. But don't worry; let me guide you down the road to code.

## Main Window

Let's first go over the code to setup the main window. The bulk of the code is in the custom view that displays our favorite word. Create a fresh Cocoa Application project to work on (don't forget to enable garbage collection). Let's dive right in and create a new `NSView` subclass called `WordView`. Make the header for `WordView` match Listing 1:

### Listing 1: WordView.h

```
#import <Cocoa/Cocoa.h>

typedef enum
{
    WordViewLeftTextAlignment,
    WordViewCenterTextAlignment,
    WordViewRightTextAlignment,
} WordViewTextAlignment;
```

```
@interface WordView : NSView
{
    NSString * _word;
    NSColor * _backgroundColor;
    WordViewTextAlignment _textAlignment;
}

@property (copy) NSString * word;
@property (copy) NSColor * backgroundColor;
@property WordViewTextAlignment textAlignment;

@end
```

This is fairly self-explanatory. We've got three instance variables and three properties for the word, background color, and text alignment. The meat is in the implementation, which is shown in full in Listing 2:

## Listing 2: WordView.m

```
#import "WordView.h"

@interface WordView ()

- (void)drawBackground;
- (void)drawWord;

@end

static NSString * RedrawContext = @"RedrawContext";

@implementation WordView

@synthesize word = _word;
@synthesize backgroundColor = _backgroundColor;
@synthesize textAlignment = _textAlignment;

- (id)initWithFrame:(NSRect)frame
{
    self = [super initWithFrame:frame];
    if (self == nil)
        return nil;

    _word = @"Word";
    _backgroundColor = [NSColor whiteColor];
    _textAlignment = WordViewCenterTextAlignment;

    [self addObserver:self forKeyPath:@"word"
              options:0 context:&RedrawContext];
    [self addObserver:self forKeyPath:@"backgroundColor"
              options:0 context:&RedrawContext];
    [self addObserver:self forKeyPath:@"textAlignment"
              options:0 context:&RedrawContext];

    return self;
}

- (void)observeValueForKeyPath:(NSString *)keyPath
                      ofObject:(id)object
                        change:(NSDictionary *)change
                       context:(void *)context
{
    if (context == &RedrawContext)
        [self setNeedsDisplay:YES];
}

- (void)drawRect:(NSRect)rect
{
    [self drawBackground];
    [self drawWord];
}

- (void)drawBackground
{
    NSRect bounds = [self bounds];

    NSRect pathRect = NSInsetRect(bounds, 2.0, 2.0);
    NSBezierPath * path =
        [NSBezierPath bezierPathWithRoundedRect:pathRect
                                        xRadius:20.0
                                        yRadius:20.0];
    [_backgroundColor set];
    [path fill];

    [path setLineWidth:4.0];
    [[NSColor blackColor] set];
    [path stroke];
}

- (void)drawWord
{
    NSRect bounds = [self bounds];
    bounds = NSInsetRect(bounds, 4.0, 4.0);

    NSFont * font = [NSFont systemFontOfSize:50];
    NSDictionary * attributes =
        [NSDictionary dictionaryWithObject:font

forKey:NSFontAttributeName];
    NSAttributedString * string =
        [[NSAttributedString alloc] initWithString:_word

attributes:attributes];

    NSSize stringSize = [string size];
    NSPoint point;
    // Center vertically
    point.y = bounds.size.height/2 - stringSize.height/2;

    // Align horizonally
    if (_textAlignment == WordViewCenterTextAlignment)
        point.x = bounds.size.width/2 - stringSize.width/2;
    else if (_textAlignment == WordViewLeftTextAlignment)
        point.x = bounds.origin.x;
    else if (_textAlignment == WordViewRightTextAlignment)
        point.x = bounds.size.width - stringSize.width;

    [string drawAtPoint:point];
}

@end
```

Inside the initializer, `initWithFrame:`, we setup initial values for the word, background color, and text alignment. We also setup key-value observers that monitor these three properties. If any of them change, we need to redraw the view, which is done by calling `setNeedsDisplay:`. The drawing itself happens inside `drawRect:` and is delegated to two methods `drawBackground` and `drawWord`.

The `drawBackground` method uses a Bezier path to create a rectangle with rounded corners. First, we fill the path with the background color, and then we stroke it with black to draw the border.

The `drawWord` method uses a class called `NSAttributedString` to draw the word with a given font and size. An `NSAttributedString` is similar to `NSString` except you can store attributes along with the string. There are many possible attributes, but we are only using the font attribute. Once we have the attributed string, we calculate the correct position inside the view and draw it with the `drawAtPoint:` method.

Remember the origin, point (0, 0), is in the lower-left corner of the view.

Now build the project, fix up any syntax errors, and open up the MainMenu.xib file in Interface Builder. Set the title of the window to Favorite Word. Next, drag a custom view from the library into the window, and set the class of the view to WordView. If you ran the application right now, it would look like Figure 4:

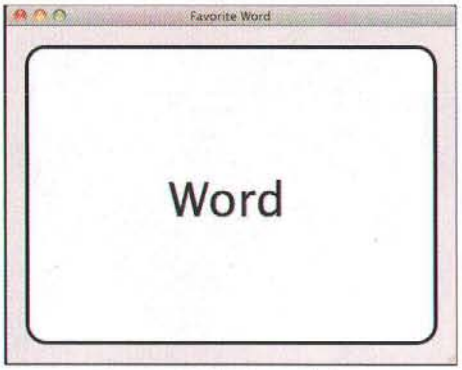



**Figure 4: Initial word view**

That's all we need for our custom view, so it's time to start getting to the meat of the matter: user preferences.

## User Preferences

• Cocoa has very good support for user preferences. The main class that provides the interface to user preferences is called `NSUserDefaults`. Every preference has a name and a value. The name must be a string, and the value must be one of the following classes:

- `NSString`
- `NSNumber` (for integers, floating point numbers and booleans)
- `NSDate`
- `NSData`
- `NSArray` or `NSDictionary` of the above classes

So let's cover how to store the favorite word in `NSUserDefaults`. For the preference name, let's use `FavoriteWord` (it's fairly customary to capitalize the name), and the value will be whatever the user supplies. Working with `NSUserDefaults` is very similar to working with a mutable dictionary, and here's how we'd set the user's favorite word to "Pie". In a real application, you wouldn't hard code the value; this is for illustration purposes only:

```
    NSUserDefaults * defaults = [NSUserDefaults
standardUserDefaults];
    [defaults setObject:@"Pie" forKey:@"FavoriteWord"];
```

That's it! The system takes care of saving this to a file periodically, so there's nothing else we need to do. Note that you use the `+standardUserDefaults` class method to get the `NSUserDefaults` instance, instead of creating a new instance of the class. This method always returns the same object and represents the defaults for your application.

Speaking of preferences files, where does the system store this file? Preferences for all applications automatically go into the directory ~/Library/Preferences. Each application has its own preference file named using its application identifier. Recall that this identifier follows the reverse DNS convention and is set in the info panel of your application. Thus, here is the full name of the preferences file for this application:

~/Library/Preferences/org.dribin.dave.mactech.jun09.Favorite_Word.plist

The extension on this file, plist, stands for *property list*. Property lists are standard file types for holding configuration information on Mac OS X. There's even a separate application for viewing and editing property lists called Property List Editor. You can use this application to verify that preferences are indeed being saved correctly, for example. Just be aware that the preferences file only exists only after a user changes a preference. It won't exist if the user only uses the standard values.

How do we read preferences using `NSUserDefaults`? That's just as easy:

```
    NSString * favoriteWord = [defaults
objectForKey:@"FavoriteWord"];
```

Let's put this newfound knowledge into practice. We're going to store the user's favorite word as a string using the `FavoriteWord` key, as we showed above. For the text alignment, we'll store the integer value of the `WordViewTextAlignment` enum, which is short for *enumrated type*. Unlike a mutable dictionary, `NSUserDefaults` has some convenience methods for storing primitive numbers so you don't have to wrap them up in an `NSNumber` yourself. Here's an example of how we save the text alignment to the preference named `TextAlignment`:

```
    NSUserDefaults * defaults = [NSUserDefaults
standardUserDefaults];
    WordViewTextAlignment alignment =
WordViewCenterTextAlignment;
    [defaults setInteger:alignment forKey:@"TextAlignment"];
```

We can read the value using the `integerValueForKey:` method.

Storing a color is a bit tricky. You'll notice that `NSColor` is not one of the supported value classes. Fortunately the `NSData` type can often be used as a catchall to handle non-standard values such as colors.

Remember that archiving allows you to turn any class that implements the `NSCoding` protocol into a stream of bytes stored in `NSData`. The `NSColor` class implements `NSCoding` so we just need to convert the color into an `NSData` before we store it in the user preferences, and then convert the `NSData` back into a color when reading out of the preferences. Here's how we'd store a red color with the `BackgroundColor` name using an `NSKeyedArchiver` to convert an `NSColor` into `NSData`:

```
    NSUserDefaults * defaults = [NSUserDefaults
standardUserDefaults];
    NSColor * color = [NSColor redColor];
    NSData * colorData =
        [NSKeyedArchiver archivedDataWithRootObject:color];
    [defaults setObject:colorData forKey:@"BackgroundColor"];
```

# Brush Your Dog's Teeth Lately?

Replaced your smoke alarm batteries? Copied your files to the server? Laptop backed up?

Despite the best of intentions, there are some things we just don't do often enough. And when comes to backup, your business continuity is at risk.

CrashPlan PRO is the first and only backup solution that combines an extremely people-friendly client with a sophisticated enterprise server to continuously protect your business onsite, offsite, and online.

Mission critical data on remote laptops, desktops and servers are backed up in real-time to multiple destinations regardless of location.

Try CrashPlan PRO in a free 30-day trial and make life easier for you, your users, and their dog's teeth.

People Friendly. Enterprise Tough.
www.crashplanpro.com

CRASHPLANPRO™
Continuous Backup for Business.

That's definitely a bit more cumbersome than storing a string, as above, but it's not too bad. Conversely, turning the data back into a color requires using `NSKeyedUnarchiver`:

```
    NSData * colorData = [defaults
objectForKey:@"BackgroundColor"];
    NSColor * color =
        [NSKeyedUnarchiver unarchiveObjectWithData:colorData];
```

Let's integrate this into our application. Create a new `NSWindowController` subclass called `MainWindowController`. Add an outlet to a `WordView` instance, as shown in Listing 3.

### Listing 3: MainWindowController.h

```
#import <Cocoa/Cocoa.h>

@class WordView;

@interface MainWindowController : NSWindowController
{
    WordView * _wordView;
}

@property (nonatomic, retain) IBOutlet WordView * wordView;

@end
```

The corresponding implementation file is shown in Listing 4:

### Listing 4: MainWindowController.m

```
#import "MainWindowController.h"
#import "WordView.h"


NSString * FavoriteWordKey = @"FavoriteWord";
NSString * BackgroundColorKey = @"BackgroundColor";
NSString * TextAligmentKey = @"TextAlignment";

@interface MainWindowController ()

- (void)updateFromDefaults:(NSNotification *)notification;

@end

@implementation MainWindowController

@synthesize wordView = _wordView;

- (void)awakeFromNib
{
    [self updateFromDefaults:nil];

    NSNotificationCenter * defaultCenter =
        [NSNotificationCenter defaultCenter];
    [defaultCenter addObserver:self
                      selector:@selector(updateFromDefaults:)

name:NSUserDefaultsDidChangeNotification
                        object:nil];
}

- (void)updateFromDefaults:(NSNotification *)notification
{
    NSUserDefaults * defaults = [NSUserDefaults
standardUserDefaults];
    _wordView.word = [defaults objectForKey:FavoriteWordKey];

    NSData * colorData = [defaults
objectForKey:BackgroundColorKey];
    NSColor * color =
        [NSKeyedUnarchiver unarchiveObjectWithData:colorData];
    _wordView.backgroundColor = color;

    WordViewTextAlignment alignment =
        [defaults integerForKey:TextAligmentKey];
    _wordView.textAlignment = alignment;
}

@end
```

The `awakeFromNib` method first updates our word view with the values stored in the preferences. But it also subscribes to `NSUserDefaultsDidChangeNotification`. This allows us to keep up-to-date if the preferences change after the application launches and will be important once we implement the preferences window.

The `updateFromDefaults:` method uses string constants instead of string literals. This helps reduce simple typo errors when using the same string over and over. The compiler will not let you use a mistyped constant, whereas a mistyped string literal can cause hard to find bugs.

If we ran the application right now, we'd run into a bit of a problem. The first time the user runs the application, their preferences are empty, and so we're not going to get any useful values out of them. What we'd like to do is setup some sensible defaults that the user can later override. We can do this by adding one more method to our implementation:

```
+ (void)initialize
{
    NSMutableDictionary * defaultValues =
        [NSMutableDictionary dictionary];
    [defaultValues setObject:@"Cocoa" forKey:FavoriteWordKey];

    NSColor * color = [NSColor redColor];
    NSData * colorData =
        [NSKeyedArchiver archivedDataWithRootObject:color];
    [defaultValues setObject:colorData
forKey:BackgroundColorKey];

    NSNumber * alignmentNumber =
        [NSNumber numberWithInt:WordViewCenterTextAlignment];
    [defaultValues setObject:alignmentNumber
forKey:TextAligmentKey];

    NSUserDefaults * defaults = [NSUserDefaults
standardUserDefaults];
    [defaults registerDefaults:defaultValues];
}
```

The `+initialize` method is a class method, not an instance method. It is also special in that it gets called automatically before the class is ever instantiated, even before `awakeFromNib`. We're using this as an opportunity to register sensible defaults with `NSUserDefaults` before `awakeFromNib` ever gets called.

Note that the `registerDefaults:` method takes a dictionary. Thus, we have to convert the alignment enum into an `NSNumber`, first. Other than that, we've setup the default favorite word to be "Cocoa", the background to be red, and have centered alignment. If we ran the application right now (don't forget to hookup the `wordView` outlet), it will look just like Figure 1 above.

# If you have a smartphone, we can sync it.


The Missing Sync product family connects the coolest devices with Mac OS X. Reliably synchronize your address book, calendar, notes, music, pictures and more between your smartphone and your Mac.*

**visit www.markspace.com/reliable**

*Available for Blackberry, Windows Mobile, Palm OS, Apple iPhone, and Symbian OS devices.
Smartphone features vary from model to model, so synchronization features and capabilities will vary from product to product.
The Missing Sync is a registered trademark of Mark/Space, Inc.

mark|space

## Preferences Window

Now that we've got our view all setup and tracking user defaults, we need to have a way for the user to actually edit them. On the one hand, this is fairly easy with Cocoa bindings. On the other hand, creating a preferences window that works like a standard preference window is not trivial.

Start off by creating a new **Window XIB** file from Xcode and call it **Preferences.xib**. Then, create a new corresponding window controller named `PreferencesWindowController`. Override the initializer to use the preferences window nib:

```
- (id)init
{
    self = [super initWithWindowNibName:@"Preferences"];
    return self;
}
```

Back in our main window controller add this action method:

```
- (IBAction)showPreferencesWindow:(id)sender
{
    if (_preferencesWindowController == nil)
    {
        _preferencesWindowController =
            [[PreferencesWindowController alloc] init];
    }
    [_preferencesWindowController showWindow:self];
}
```

You'll also need to add the corresponding instance variable to the header. This action method uses the preferences window controller to load and display a preferences window. The menu that you want to connect this to is named **Preferences...** under the application's menu, as shown in Figure 1.

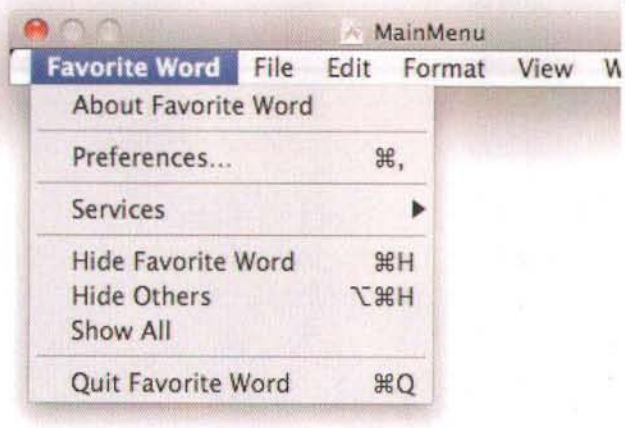



**Figure 5: Preferences menu**

We are going to want to customize many of the window attributes of the preferences window. Make sure they all match those in Figure 6.

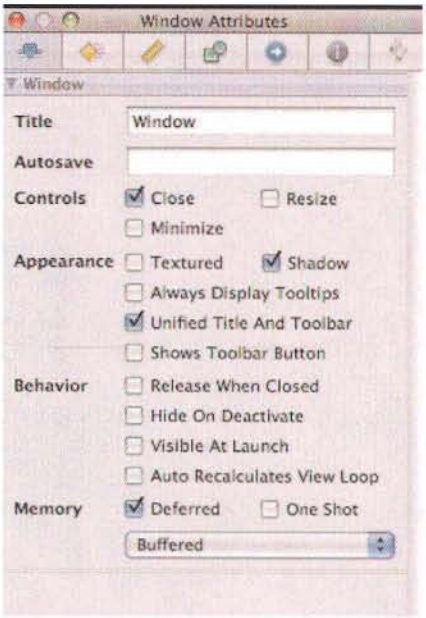



**Figure 6: Preferences window attributes**

We now have enough in place that you can test the preferences window. It currently doesn't do anything useful, but you can make sure the **Preferences...** menu is hooked up properly and displays the preferences window from the nib file.

The first step in creating a standard preferences window is to add a toolbar to this window. Toolbars are typically used to add shortcuts to commonly used actions, but they are also what give preferences windows their distinctive look.

Drag a toolbar out from the **Library** and onto your preferences window. It comes preconfigured with some standard toolbar items, and while these may be useful for a traditional toolbar, we don't want any of them for our preferences window. Double click on the toolbar and a customize sheet appears, as in Figure 7.

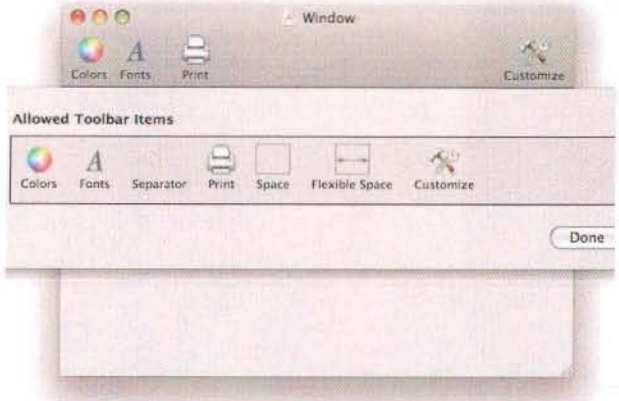



**Figure 7: Default toolbar**

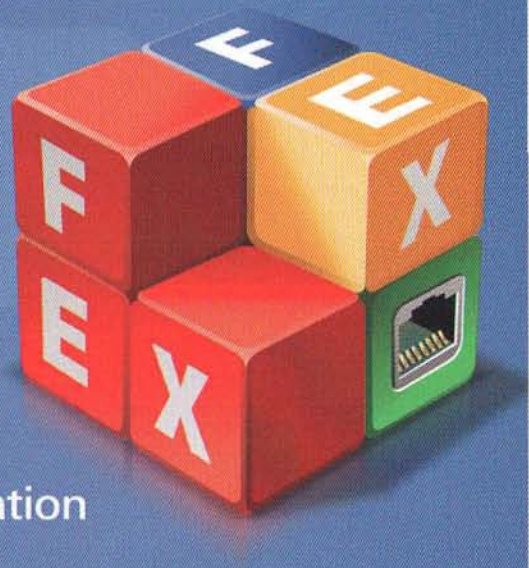
Drag each and every toolbar item off the **Allowed Toolbar Items** section to get an empty toolbar. Replace them with two **Image Toolbar Items** from the **Library**. Configure the first one on

# ESC> the pain of font management

ESC> frequent font server crashes

ESC> font licensing headaches

ESC> limited font selection for creatives




ESC> unintuitive font management administration

ESC> the tech support runaround

ESC> multiple contacts for font management, font purchases, font license management and other font needs

**ESC> the pain of font management with the FontExplorer® X Server solution**

**FontExplorer X: font management from the font experts**

Linotype and Monotype Imaging have teamed to bring you FontExplorer X Server. This unmatched font management solution combines a robust feature set that the creative workflow demands, a solid architecture and an intuitive nature that network administrators need, and a selection of world class fonts that creatives crave. With powerful font management capabilities and licensing for our renowned Linotype®, Monotype® and ITC® OpenType® font libraries, FontExplorer X Server provides an unprecedented solution for managing fonts across creative organizations or large design groups. See how your organization can benefit from centralized font management — and the convenience of having a single contact for font management, font licensing, font license management and any other typographic request.

**Call us at (800) 424-8973 for a free evaluation.**

Monotype Imaging

**Monotype Imaging Inc.**
500 Unicorn Park Drive
Woburn, MA 01801
Toll Free: (800) 424-8973
Phone: 781 970 6000

Linotype

**Linotype GmbH**
Werner-Reimers-Straße 2-4
61352 Bad Homburg
Germany
Phone: +49 (0) 6172 484-418

**www.fontexplorerx.com**

the left to have the attributes in Figure 8. Set the Image Name to NSPreferencesGeneral, both the Label and Pal. Label to General, and the Tag to 0.

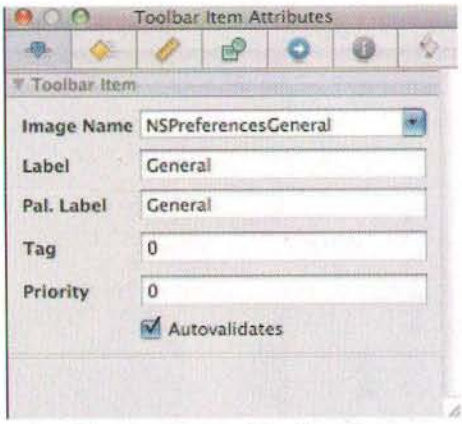



**Figure 8: General toolbar item**

Configure the second toolbar item similarly, setting the Image Name to NSAdvanced, the Label and Pal. Label to Advanced and the Tag to 1, as show in Figure 9.

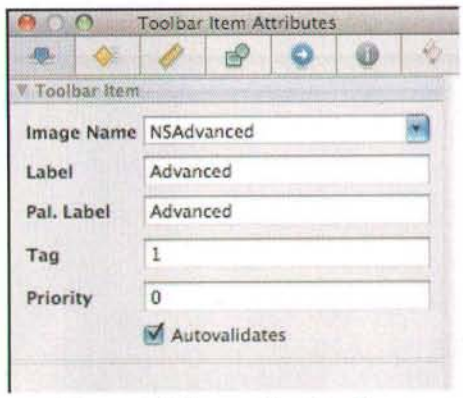



**Figure 9: Advanced toolbar item**

Drag each toolbar item from the Allowed Toolbar Items section onto the actual toolbar, and your window should look like Figure 10.

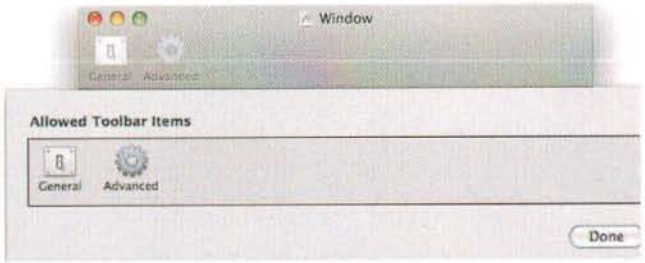



**Figure 10: Preferences toolbar**

We're done editing the toolbar for now (we'll have to come back and connect actions to the items later), so click on the Done button. Edit the attributes of the toolbar itself to match Figure 11, which should just be unchecking the Customizable checkbox.

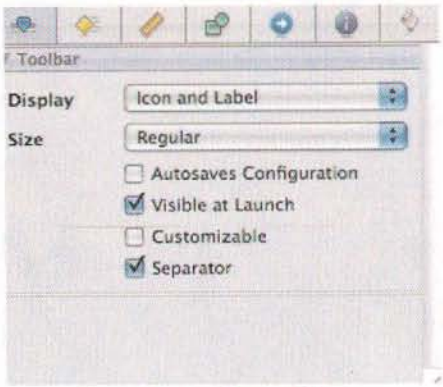



**Figure 11: Toolbar attributes**

One last thing before jumping back to Xcode, set the class of File's Owner to be the PreferencesWindowController and set the delegate of the toolbar to be File's Owner. Also, I want to point out that editing toolbars and toolbar items is new to Interface Builder in Mac OS X 10.5 In previous versions of Mac OS X, you had to create the toolbar and toolbar items all in code.

Back in Xcode, add this toolbar delegate method:

```
- (NSArray *)toolbarSelectableItemIdentifiers:(NSToolbar
*)toolbar
{
    NSMutableArray * identifiers = [NSMutableArray array];
    for (NSToolbarItem * item in [toolbar items])
    {
        [identifiers addObject:[item itemIdentifier]];
    }
    return identifiers;
}
```

Normally, toolbar items work like push buttons: they are only highlighted when the mouse is down. Selectable toolbar items stay highlighted after the mouse is clicked and are drawn with a special highlight. Our method tells the toolbar that *all* items are selectable.

## View Controllers

Before we finish off the rest of the code for the preferences window controller, let's talk about what we're going to accomplish. Open the preferences for a standard Apple application, such as Mail, iCal, or Address Book. You'll notice that when you click on a toolbar item, the contents of the window are briefly blanked until the window resizes and the contents of the window are replaced with new controls. If you watch closely, you'll notice that the window only resizes vertically. The width stays the same, no matter which preference pane is selected. What's happening is a technique called *view swapping*.

We're going to put our General preference pane and Advanced preference pane into their own views. Then, when the toolbar is clicked, we're going to swap out the current view and swap in the appropriate view. As another bonus, we're going to store these views in their own nib. Just like keeping windows in their own nib, storing views in their own nib reduces memory

consumption by only loading the views as they are needed. If the user never clicks on the Advanced preference pane, it is never loaded into memory.

Just as we use a window controller to load a window from a nib file, there is a class new to Mac OS 10.5 called a *view controller* that loads a view from a nib file. Let's create our view and view controller for our General preferences pane.

In Xcode, create a new class, name it `GeneralPreferencesController`, and change the super class to `NSViewController`, as shown in Listing 5.

### Listing 5: GeneralPreferencesController.h

```
#import <Cocoa/Cocoa.h>

@interface GeneralPreferencesController : NSViewController
{
}

@end
```

The implementation class is short, as shown in Listing 6.

### Listing 6: GeneralPreferencesController.m

```
#import "GeneralPreferencesController.h"

@implementation GeneralPreferencesController

- (id)init
{
    self = [super initWithNibName:@"GeneralPreferences"
bundle:nil];
    return self;
}

@end
```

All it does is load the correct nib file. You could argue that a separate subclass is not worthwhile in this case, and that's probably true. But real preference panes will most likely need extra code behind them for actions and outlets, so you'd need to create a subclass at that point. We're lucky enough to be able to use Cocoa bindings, but I think it's a good idea to create the subclass up front so you have a place to put code when you need it.

Now create the corresponding nib file by creating a new View XIB file, as shown in Figure 12. Name this nib file GeneralPreferences.xib, and open it up in Interface Builder.

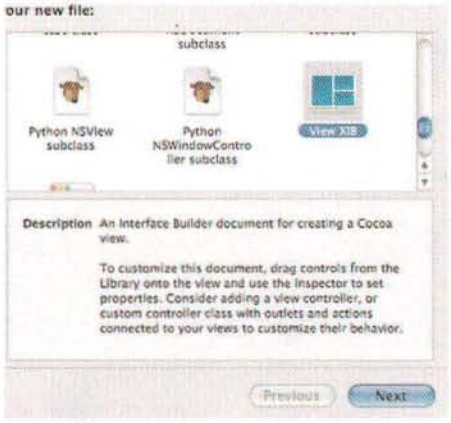



**Figure 12: New View XIB file**

The first thing you'll want to do is set the File's Owner class to GeneralPreferencesController. Next, you'll want to add a label and a text field to the view. Note that in Interface Builder, our view looks an awful lot like a window. But keep in mind that, despite its looks, it's a bare view without an enclosing window. The final layout should look like Figure 13.

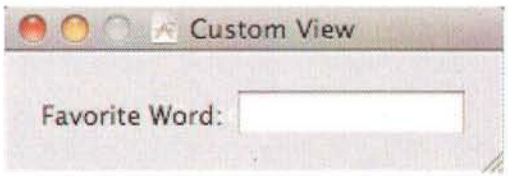



**Figure 13: General preferences view**

Using Cocoa bindings, we can keep this text field in sync with `NSUserDefaults` without writing any code. Open up the Bindings section of the Inspector panel for the text field, and bind to Shared User Defaults Controller, setting the Controller Key to values and the Model Key Path to FavoriteWord (with no space) as shown in Figure 14.

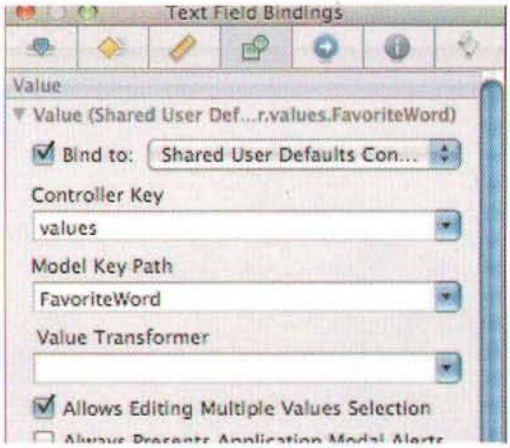



**Figure 14: Word view bindings**

The Shared User Defaults Controller is a special, built-in object controller that connects directly to the shared `NSUserDefaults`. The model key path is the name of the preference you want to bind to, so this must be the same string we used in the main window controller. And through the magic of bindings, we've successfully allowed the user to edit their favorite word.

We now have to go through similar steps for the Advanced preference pane. Create a new view controller subclass, but this time name it `AdvancedPreferencesController`. Override the initializer and load the nib file named `AdvancedPreferences`. Finally, create a new View XIB file named AdvancedPreferences.xib and open this in Interface Builder.

Again, the first step is to change the File's Owner class to be `AdvancedPreferencesController`. Layout the view to match Figure 15 by dragging two labels, a color well, and a radio button

group from the Library onto the view. By default, a radio group only has two buttons. To create the third button, drag down as if you were resizing the view, but hold down the Option key.

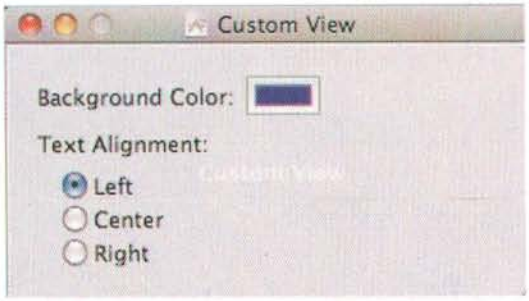



**Figure 15: Advanced preferences view**

Again, we can connect the color well and radio button group using Cocoa bindings. For the color well, bind the Value to the Shared User Defaults Controller, but this time use BackgroundColor as the Model Key Path. We also have to deal with the fact that the color is stored in the preferences as `NSData`. Change the Value Transformer to be NSKeyedUnarchiveFromData. Value transformers act as a middleman between the view and the controller. There are various built-in transformers, and you can create your own, but we can use the one that archives and unarchives the value. The bindings options should match Figure 16.


**Figure 16: Color well bindings**

For the radio button group, you are going to bind the Selection Indexes to Shared User Defaults controller. Set the Model Key Path to TextAlignment. There's no need to change anything else, as it will automatically convert to and from an `NSNumber` instance.

Make sure both view nibs are saved, and it's time to head back into Xcode to code up the view swapping. Update the header file for `PreferencesWindowController` to match Listing 7. We've added two instance variables, one for each view controller, and an action method that the toolbar items will use.

## Listing 7: PreferencesWindowController.h

```
#import <Cocoa/Cocoa.h>

@class GeneralPreferencesController;
@class AdvancedPreferencesController;


@interface PreferencesWindowController : NSWindowController
{
    GeneralPreferencesController * _generalPreferences;
    AdvancedPreferencesController * _advancedPreferences;
}

- (IBAction)changePreferencePane:(id)sender;

@end
```

In the implementation file, add the accessors for the view controllers:

```
- (GeneralPreferencesController *)generalPreferences
{
    if (_generalPreferences == nil)
    {
        _generalPreferences =
            [[GeneralPreferencesController alloc] init];
    }
    return _generalPreferences;
}

- (AdvancedPreferencesController *)advancedPreferences
{
    if (_advancedPreferences == nil)
    {
        _advancedPreferences =
            [[AdvancedPreferencesController alloc] init];
    }
    return _advancedPreferences;
}
```

These create the objects as needed. Again, this keeps memory consumption down by only creating objects when they are needed. Next, add these three methods that implement the view swapping:

```
- (IBAction)changePreferencePane:(id)sender
{
    [self selectPreferencesForItem:sender animate:YES];
}

- (void)selectPreferencesForItem:(NSToolbarItem *)item
                         animate:(BOOL)animate
{
    NSInteger tag = [item tag];
    NSViewController * preferencesController = nil;
    if (tag == PreferencesGeneralTag)
        preferencesController = [self generalPreferences];
```

# What Apps do your friends have?

AppSpace.com

ZAGG™ Home of the invisible SHIELD® by ZAGG.



NASDAQ BB: ZAGG

```
    else if (tag == PreferencesAdvancedTag)
        preferencesController = [self
advancedPreferences];

    [self selectPreferences:preferencesController
animate:animate];
    [[self window] setTitle:[item label]];
}

- (void)selectPreferences:(NSViewController *)preferences
                  animate:(BOOL)animate
{
    NSView * contentView = [[self window] contentView];
    NSView * preferencesView = [preferences view];

    // Calculate the change in height
    NSSize currentSize = [contentView frame].size;
    NSSize newSize = [preferencesView frame].size;
    CGFloat deltaHeight = newSize.height -
currentSize.height;

    // Calculate the window's new frame
    NSWindow * window = [self window];
    NSRect windowFrame = [window frame];
    windowFrame.size.height += deltaHeight;
    windowFrame.origin.y -= deltaHeight;

    // Remove the current view
    for (NSView * view in [contentView subviews])
        [view removeFromSuperview];

    // Resize the window
    [window setFrame:windowFrame display:YES
            animate:animate];

    // Resize the new view's width
    newSize.width = currentSize.width;
    [preferencesView setFrameSize:newSize];

    // Add it to the window
    [contentView addSubview:preferencesView];
}
```

Let's work our way through these methods from the top down. The first method is our action method that gets called when either of the toolbar items is clicked. The sender of the action will be the toolbar item that the user clicked. This simply calls into the `selectPreferencesForItem:` method with the `animate` argument set to `YES`.

The second method uses the tag of the toolbar item to select the correct view controller. We use an enum to map the tag values into compile time constants. This ultimately calls through to the third method, `selectPreferences:animate:`, which does the actual view swapping. After the view swapping is finished, it sets the title of the window to be the same as the toolbar item label.

The algorithm for view swapping is fairly simple: remove the existing view, resize the window with or without animation, and add in the new view. The only tricky part is knowing how much to resize the window. We compute the difference in height between the current view and the view we are swapping to, and change the frame of the window by that same amount. Remember, the origin is in the lower-left, again, so we need to adjust the origin so that the top of the window does not move. The `setFrame:display:animate:` method does the fancy animation for us. All we need to do is remove the current view before resizing the window and add in the new view when it's finished. We also ensure the new view's width is resized to the width of the window.

That's the bulk of it. We need to make sure to connect up the toolbar items to the `changePreferencePane:` action and add in two more methods for some final touches:

```
- (void)showWindow:(id)sender
{
    NSWindow * window = [self window];
    if (![window isVisible])
        [window center];

    [super showWindow:sender];
}

- (void)windowDidLoad
{
    NSToolbar * toolbar = [[self window] toolbar];
    NSToolbarItem * firstItem = [[toolbar items]
objectAtIndex:0];
    [toolbar setSelectedItemIdentifier:[firstItem
itemIdentifier]];
    [self selectPreferencesForItem:firstItem animate:NO];
}
```

The first method overrides the default implementation of `showWindow:` to center the window on screen before displaying it. This is not strictly necessary, but I find it looks nicer. The `windowDidLoad` method is necessary to ensure that the General preferences view is initially swapped in. Notice that we're using the same method the toolbar action method uses, but we're setting `animate` to `NO`, as we want the window to display immediately without any animation.

## Conclusion

This has probably been the longest example we've done so far. If you don't want to type in all this code, feel free to download the completed project from the MacTech website. Congrats for keeping up. More goodies to come next month in *The Road to Code*.


### About The Author

***Dave Dribin has been writing professional software for over eleven years. After five years programming embedded C in the telecom industry and a brief stint riding the Internet bubble, he decided to venture out on his own. Since 2001, he has been providing independent consulting services, and in 2006, he founded Bit Maki, Inc. Find out more at <http://www.bitmaki.com/> and <http://www.dribin.org/dave/>.***

# Scripting Opportunities for System Administrators, Part One

## When, where, why, and how you should run administrative scripts

*By Greg Neagle, MacEnterprise.org*

**MacEnterprise.org**
Mac OS X enterprise deployment project

## Introduction

In previous MacTech columns, I have sometimes offered up a script or two as part of a solution to a particular problem. Even if the script I presented is a perfect fit for your environment, you might still have had trouble making effective use of it in your organization, because you did not know how to make it run at the right time or in the correct context to get the job done.

This month, we'll begin a look at some of the many mechanisms available to run scripts (and other processes). Each mechanism has different uses and is suited for a unique set of tasks. Depending on what you need to accomplish, you should select the appropriate mechanism.

## Why?

The first question you should ask is "Why do I want to run this script? What task do I want to accomplish?" Some of the common administrative categories that might lend themselves to scripting are:

System configuration – initial setup of the OS, networking, user accounts, etc.

System management – ongoing management of system settings; enforcing system-wide policies

User settings/preferences – configuring applications; setting up useful default preferences, enforcing user-level policies

Administrator tools – tools to make tedious or difficult tasks easier or more consistent

Hacks/fixes/workarounds – scripts to "fix" or work around problems with the OS or applications (or users!)

## When?

Once you know why you are running the script, or what you want to accomplish with the script, you can consider when it should run. Some of the possibilities:

On demand – only when invoked by an admin or user

At startup

Repeating (daily/weekly/monthly, or other intervals like hourly, every 15 minutes, etc)

When a user logs in

When a user logs out

Let's look at some logical pairings of "Why?" and "When?":

Utility scripts or administrative tools that are to be run only on demand are the easiest to handle. Simply put them in a directory somewhere and run them manually as needed, or run them remotely using Apple Remote Desktop or SSH.

If your script is doing system configuration tasks like binding a machine to Active Directory or creating local users, it should almost certainly run at startup. System management tasks might run at startup, on a repeating schedule, or both.

Tasks that affect user settings or preferences should probably run at user login, or if it's a cleanup task, perhaps at user logout.

Hacks/fixes/workarounds can vary when it is appropriate to run them: they may need to run on demand; they may need to run at login to make a change to a user's environment; they may need to run at startup to clear out stale cached data.

## How?

How can you run your script when you want? Let's look at some of the available mechanisms.

## Running a script at startup

There are several ways to run scripts at startup, but the two most commonly used on Mac OS X are **StartupItems** and **launchd** items. Both Startup Items and launchd's LaunchDaemons run in the root context.

### StartupItems

Startup items have been used on Mac OS X for a long time, and continue to work in OS X 10.5 Leopard. Administrator-provided Startup items should be placed in `/Library/StartupItems`. Since Apple is phasing out StartupItems, we won't spend a lot of time on these. A Startup Item consists of a directory containing an executable (typically a shell script) and a `StartupParameters.plist` file. Both the executable/shell script and the `StartupParameters.plist` have a very specific format. The startup item directory may optionally contain other items – for example, a Resources subdirectory. Apple has some good documentation on creating StartupItems here:

http://developer.apple.com/documentation/MacOSX/Conceptual/BPSystemStartup/Articles/StartupItems.html

### launchd items

`launchd` was introduced with Mac OS X 10.4 Tiger. It was intended to replace almost every other way of launching processes, though in practice it has not yet accomplished that. Still, `launchd` is very capable, and Apple continues to enhance it. `Launchd` items are simpler to setup than StartupItems, as they generally require only a single additional file other than the actual script or executable. Administrator-provided `launchd` items that run a script at startup should go in /Library/LaunchDaemons.

`Launchd` has been covered extensively in MacTech as well as many other places, so I won't go into great depth. But let's do a quick example. Let's say we have a configuration script that we want to run at startup. It is located at `/Library/Management/configuration.sh`, and is marked as executable. It looks like this:

```
#!/bin/sh
logger -t configuration "Hello from the config script!"
```

This script simply writes a message to the system log. You can test it manually:

```
root# /Library/Management/configuration.sh
root# tail /var/log/system.log
<snip>
Apr 21 12:31:05 arcus configuration[8400]: Hello from the
config script!
```

To get it to run at startup, you'd need a property list at `/Library/LaunchDaemons/com.mactech.demo.plist` that looks something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST
1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
```

```
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>com.mactech.demo</string>
    <key>ProgramArguments</key>
    <array>

<string>/Library/Management/configuration.sh</string>
    </array>
    <key>RunAtLoad</key>
    <true/>
    <key>OnDemand</key>
    <true/>
</dict>
</plist>
```

This property list should have owner: root, group: wheel, and mode: 0644. Some explanation of the included keys:

**Label** is a string used by **launchd** to identify the job. The name of the plist file is usually the same as this label plus ".plist".

**ProgramArguments** is simply the full path to the executable script.

**RunAtLoad** is set to **true** so that **launchd** will run the job when it loads it, which is normally at startup.

**OnDemand** is set to **true** so that **launchd** won't attempt to restart the script once it exits – in other words, this process/script is not meant to run continuously.

We could test the launchd job by rebooting and then looking at the system log for our message, but rebooting can take a while, and if there's a problem, the fix/retest cycle is tedious. So let's do a quicker test:

```
root# launchctl load
/Library/LaunchDaemons/com.mactech.demo.plist
root# tail /var/log/system.log
<snip>
Apr 21 12:31:05 arcus configuration[8400]: Hello from the
config script!
Apr 21 12:37:13 arcus configuration[9073]: Hello from the
config script!
```

At the end of the system log, you should see the message from the configuration script.

If you have problems, unload the job:

```
root# launchctl unload
/Library/LaunchDaemons/com.mactech.demo.plist
```

Make your changes to the plist, and try loading the job again. Once the job is working, you should be able to reboot, and see the message in the system log during the startup process.

Apple documentation on creating a **launchd** item is available here:

http://developer.apple.com/documentation/MacOSX/Conceptual/BPSystemStartup/Articles/LaunchOnDemandDaemons.html

## Repeating scripts

Some scripts are best run on repeating intervals. For example, you have a script that scans the startup disk for all installed fonts and then uploads that list of fonts to a database somewhere so you can monitor for license compliance. You should run that script periodically: maybe daily, maybe weekly, maybe monthly—it's up to your organization. There are several ways to do this.

One of the easiest ways is to piggy-back off an existing facility for running repeating scripts: the **periodic** command. By default, **periodic** is used to run scripts on a daily, weekly or monthly basis. It runs all the scripts it finds in certain directories:

**/etc/periodic/daily/** – these are run every day

**/etc/periodic/weekly/** – these are run once a week

**/etc/periodic/monthly/** – these are run once a month

To get periodic to run your scripts, mark them as executable and put them in the appropriate directory. You can control the order in which the scripts run by naming them appropriately. The convention used is to start the script name with a three-digit number; the scripts are then run in numeric order:

```
root# ls -1 /etc/periodic/daily
100.clean-logs
110.clean-tmps
130.clean-msgs
430.status-rwho
500.daily
599.randomSleep
600.updateMachineName
700.updateHostInfo
900.autoradmind
```

There are some issues to be aware of, however. The first is exactly when these scripts will run. This is controlled by launchd in the following files in **/System/Library/LaunchDaemons**:

```
com.apple.periodic-daily.plist
com.apple.periodic-monthly.plist
com.apple.periodic-weekly.plist
```

Looking at **com.apple.periodic-daily.plist**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST
1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>com.apple.periodic-daily</string>
    <key>ProgramArguments</key>
    <array>
        <string>/usr/sbin/periodic</string>
        <string>daily</string>
    </array>
    <key>LowPriorityIO</key>
    <true/>
```

# JILLIAN'S
AT METREON

# The Perfect Place to Host your Next Event!

## The Main Dining Room

- 500 Guests Reception Style
- 175 Guests for Plated Lunch or Dinner
- 300 Guests Buffet Style
- A Great Spot for Off-Site Company Meetings
- 50-foot Video Wall and Bar
- State of the Art A/V Hook-ups

## The Billiard Lounge

Featuring 10 Brunswick Gold Crown Tournament Style Billiards Tables and Can Accommodate:

- 150 Guests Reception Style
- 120 Guests Plated Meal
- Can Be Sectioned Off for Smaller Gatherings With Use of the Outdoor Patio

## The Martini Bar

- Up to 50 Guests Reception Style
- 35 Guests Buffet Style
- Up to 40 Guests for Sit-Down Events
- Complete With a Plasma Screen TV
- Wireless Internet Connection

## The Lower Dining Room Can Accommodate:

Up to 80 Guests for Sit-Down Events

100 Guests for Receptions – 75 Guests for Theater-Style Events

Contact our Event Sales Manager:
415.369.6101/ ecsanfrancisco@jilliansbilliards.com

```
    <key>Nice</key>
    <integer>1</integer>
    <key>StartCalendarInterval</key>
    <dict>
        <key>Hour</key>
        <integer>3</integer>
        <key>Minute</key>
        <integer>15</integer>
    </dict>
</dict>
</plist>
```

The `StartCalendarInterval` key tells us it will run each day at 3:15 AM local time. That time might not be a good one for your environment. Consider a few scenarios:

Desktop machines: if the desktops in your organization are left on 24/7, then the script will run shortly after 3:15 AM each day. If instead, they are asleep at 3:15 AM, they'll run shortly after they are woken up each day. If they are powered off at 3:15 AM, the scripts won't run at all. Launchd will reschedule jobs that were scheduled to run when the machine was asleep, but will not reschedule jobs that were scheduled to run when the machine was powered off.

Laptop machines: all of the same complications as with desktops, with the additional problem that if they get taken home at night or over the weekend, and run these jobs at 3:15 AM (or when woken up), and your scripts require access to network services or resources available only when the machine is connected to your organization's network, they may fail, or at the very least, fail to do anything useful.

Therefore, you might consider changing the time these jobs run to a time during the day when it's more likely the machine is on, awake, and on your network.

Another possible approach that does not require modifying Apple's provided launchd items is to run a script at startup that checks for overdue periodic jobs and runs them. Such a script is described (and available) here:

http://managingosx.wordpress.com/2008/06/18/launchd-vs-periodic/

## Repeating launchd jobs

Looking at the property list for `com.apple.periodic-daily.plist` suggests another method for running scripts on a repeating basis: avoiding the periodic middleman and using launchd directly. Using our `com.mactech.demo` example as before, we can run a script at startup and once a day with a LaunchDaemon plist like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST
1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>com.mactech.demo</string>
    <key>ProgramArguments</key>
    <array>

<string>/Library/Management/configuration.sh</string>
    </array>
    <key>RunAtLoad</key>
    <true/>
    <key>OnDemand</key>
    <true/>
    <key>StartCalendarInterval</key>
    <dict>
        <key>Hour</key>
        <integer>12</integer>
        <key>Minute</key>
        <integer>15</integer>
    </dict>
</dict>
</plist>
```

We've added a `StartCalendarInterval` to the previous version of the property list that tells `launchd` to run the job each day at 12:15, when many of our staff will be at lunch. If we didn't want it to run at startup as well, we could remove the `RunAtLoad` key, or set it to false.

## Good old cron

One last method to mention: the classic UNIX `cron` is still available in OS X, and can still be used to run repeating jobs. Type `man cron` at a command prompt for details. You'll probably want to create a `crontab` at `/etc/crontab`. A big disadvantage of using `cron` is that it's hard to manage different versions of the crontab files. For example, if you have a set of machines that need repeating jobs A and B, another set that needs repeating jobs B and C, and yet another set that needs repeating jobs A, B and C, you'll need to manage three different versions of the `crontab` as well as the scripts that do the actual jobs. If you use `periodic` or `launchd`, you don't have to deal with the monolithic `crontab` file, as the scheduling info for each job lives in a separate file (or in the case of `periodic`, is not needed).

# To be continued...

We covered quite a bit this month. We looked at running scripts at startup and on a repeating schedule. In the future, we'll look at running scripts as part of the login and logout process, both with root privileges, and as the user logging-in. While you might guess that `launchd` might be useful here, we'll also look at login/logout hooks and login items. We'll also consider the special case of scripts that should run only once, either at startup or login. Finally, we'll look at some methods to simplify implementing additional scripts once you have a few working. See you next time!


## About The Author

***Greg Neagle is a member of the steering committee of the Mac OS X Enterprise Project (macenterprise.org) and is a senior systems engineer at a large animation studio. Greg has been working with the Mac since 1984, and with OS X since its release. He can be reached at gregneagle@mac.com.***

# Installer Plugins

## Build a basic installer plug-in using Xcode

by José R.C. Cruz

## Introduction

Up to now, you know two ways to customize an install session. For instance, to check if the target is the right one for the payload, you use a requirements script. To prepare the target to receive the payload, you use an action or an install script. Yet, there are cases where you may find these approaches inadequate. To handle those cases, you may need an installer plug-in.

This article will give you the background you will need to write your own installer plug-in. First, it explains how the plug-in fits into the basic install session. Then, it describes the plug-in API defined by the `InstallerPlugin.framework`. Next, it presents the plug-in template and its constituent files. Finally, it shows how to build a basic plug-in and test it in a basic package.

As always, you can get a copy of the featured projects from the MacTech website. Just go to the following URL to download your copy.:

ftp.mactech.com/src/mactech/volume25_2009/25.06.sit

## Enter The Plug-In

Several panels make up a basic install session (Figure 1). The **Welcome** panel appears when users start the session by double-clicking the package. The second panel, **ReadMe**, provides users with more information about the package's payload. The third one, **License**, shows the manufacturer's license terms governing the use of the payload. User can either reject these terms, thus ending the session, or accept them to continue. Next, the **Select Destination** panel lets users choose the target volume on which to install the payload. After that, the **Custom Install** panel allows users to select which payload to install on the target. Then the **Standard Install** panel gives users one last chance to change their minds before the actual install occurs. Finally, the **Conclusion** panel summarizes the results of the installation.

Not all of these panels need appear in an install session. For instance, the **Readme** and **License** panels (*grey*) are optional. The install session skips these panels if they have no text to display. The **Select Destination** and **Custom Install** panels (*orange*) are controlled by the package. They appear only if users are allowed to change the target volume or choice of payloads during the session.

Now suppose your package needs extra information from the users, and none of the panels are up to the task. For this case, you need to write an installer plug-in. A plug-in can insert a custom panel at specific points of the session (Figure 2). It resides in the directory `Contents/Plugins` of the package. Also, a plug-in has full access to the Cocoa framework. This allows the plug-in to do a wide variety of tasks, tasks that are either difficult or impossible to do with a script.

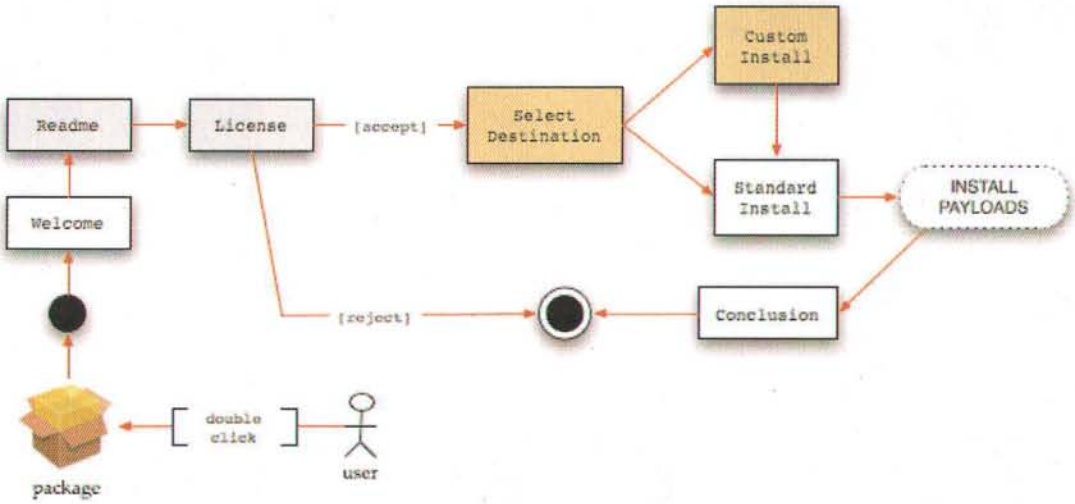



**Figure 1. Sequence of panels in a basic install session.**

Yet, like any software technology, an installer plug-in has its limits and issues. For one, it cannot replace any of the basic install panels. A plug-in must always provide a panel -- paneless plug-ins are not supported. Also, you cannot debug a plug-in using Xcode's source debugger. Your only recourse is to have the plug-in send debug messages to the `console.log` file using either `NSLog()` or ASL (Apple System Log). Another issue is that only meta-packages and distribution packages support installer plug-ins. The new flat-file Leopard package does not support them at this time.

Last of all, at the time of writing, the plug-in API is still poorly documented. Your only options so far are to study the header files in the InstallerPlugin framework, and the sample plug-in project from Apple.

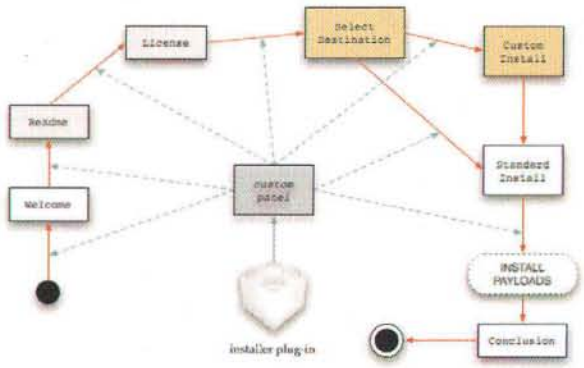



**Figure 2. A plug-in in an install session**

# The Plug-in Framework

The InstallerPlugin framework serves as the basis of all installer plug-ins. This framework is located in `/System/Library/Frameworks` of the OS X boot volume. The framework comes with four header files, one of which, `InstallerPlugins.h`, is the main header. The other headers define three classes your plug-in can use. Figure 3 shows how these classes relate to each other.

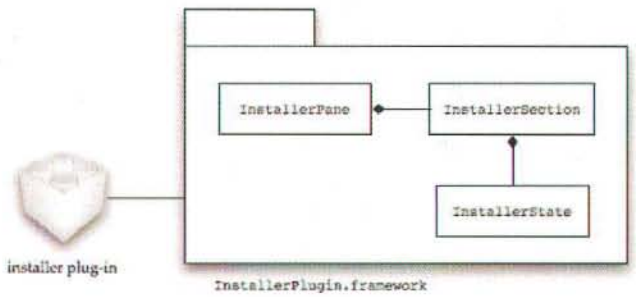



**Figure 3. The InstallerPlugin framework**

To add this framework to your Xcode project, select the Frameworks and Libraries group on the Groups & Files pane of the editor window. Choose Add to Project from the Project menu, and use the Open File dialog to select the framework. Click the Add button to include the framework to the project. Then add the following line to your header file

```
#import <InstallerPlugins/InstallerPlugins.h>
```

Let us now examine what each class has to offer.

## The InstallerPane class

The InstallerPane class (Figure 4) handles the display of the custom panel. It also manages the interactions between the panel and the users. In short, this class serves as the controller for that panel.

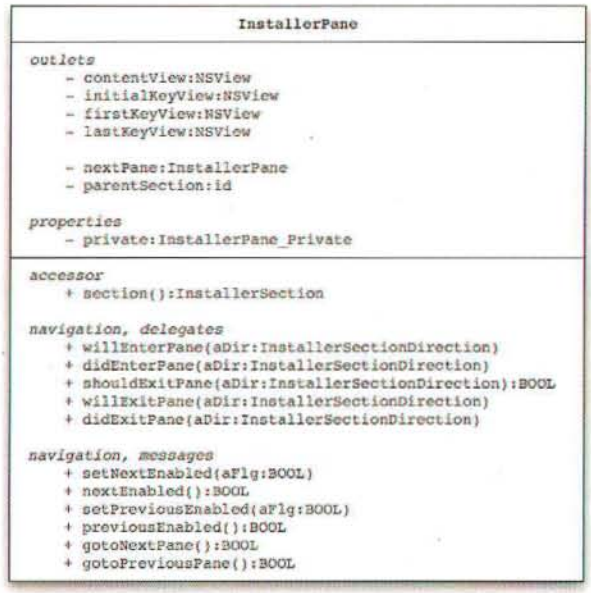



**Figure 4. The InstallerPane class**

The class has six private outlets, four of which give the views that are linked to the class. For instance, the `contentView` outlet is the panel itself. The `initialKeyView` outlet is the first control widget that gets user focus after the panel is displayed. The `firstKeyView` outlet points to the current widget that gets any keyboard events, while the `lastKeyView` outlet points to the last widget to get any events.

The `nextPane` outlet returns the installer panel that follows the current one. And the outlet `parentSection` returns the InstallerSection instance for that panel (more on this later).

The InstallerPane class also comes with a wide range of messages. In this article, we will focus only on those messages that deal with panel behavior. There are eleven of these messages, which falls under two groups. The first group consists of delegate messages that the class gets at each panel event (Figure 5).

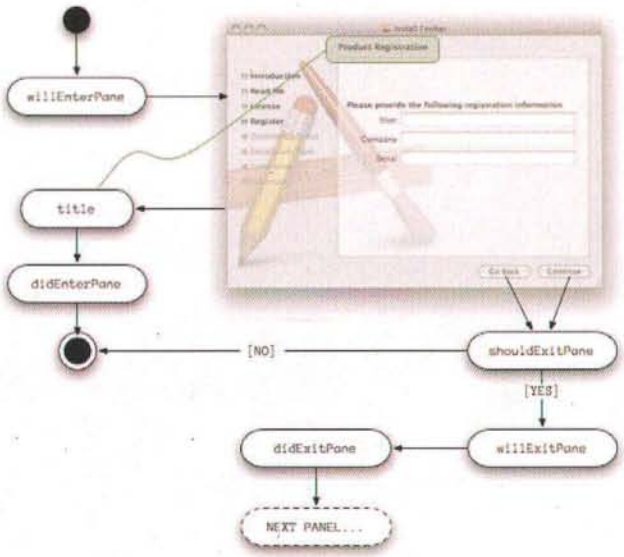



**Figure 5. Panel events and delegate messages**

Before the panel appears in the install session, it sends a `willEnterPane` message to the InstallerPane class. This allows the class to prepare the resources it needs to support the panel. Next, the panel appears and sends a `title` message to the class. The class responds with a localized NSString, which the panel displays near its upper-right corner (*green*). Then the panel sends a `didEnterPane` to the class. The latter can respond either by setting the default values on the panel or by starting the desired services.

When users click the Go Back or Continue button, the panel first sends a `shouldExitPane` to the InstallerPane class. If the class returns a `NO`, the panel remains active. On the other hand, if the class returns a `YES`, the panel sends a `willExitPane` back to the class. The class uses this moment to process the user data from the panel. The panel then disappears and sends a `didExitPane` back to the class. This is where the class can dispose the resources it used to support the panel.

The second group of messages allows the InstallerPane class to control some panel activity. For instance, to enable the Continue button, send a `setNextEnabled` message with a `YES` argument.

```
[self setNextEnabled:YES];
```

To read the state of the Continue button, send a `nextEnabled` message. If the button is enabled, the message returns a `YES`; otherwise, it returns a `NO`.

```
tFlg = [self nextEnabled];
```

To display the next panel, use the `gotoNextPane` message. For the previous panel, use the `gotoPreviousPane` message.

```
tFlg = [self gotoNextPane];
```

These messages have the same effect as users clicking the Continue or Go Back buttons. Both return a `YES` if the desired panel appears without errors. On the other hand, if the panel does not exists or if an error occurs, both messages return a `NO`.

## The InstallerSection class

Next is the InstallerSection class (Figure 6), which works as a controller for InstallerPane. It supplies the plug-in with data on the current install session. The InstallerPane class carries an instance of InstallerSection in its `parentSection` outlet. To access the instance, send a `section` message from InstallerPane.

```
tSct = [self section];
```

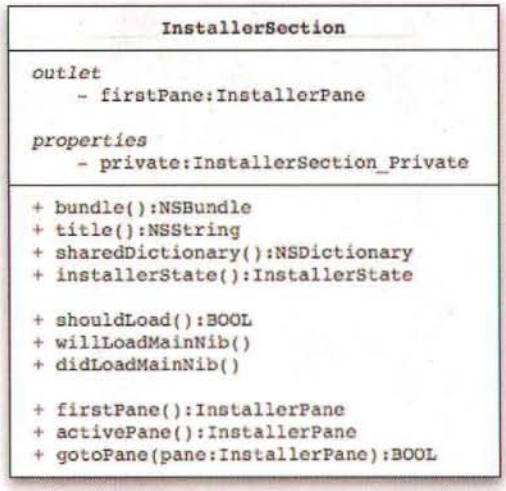



**Figure 6. The InstallerSection class**

The InstallerSection class has one private outlet, `firstPane`. This outlet stores an instance of the InstallerPane class. By default, this is the same InstallerPane whose the custom panel appears during the install session. Since this outlet exists, it implies that a plug-in can have multiple instances of InstallerPane, each one with its own custom panel. We will explore this possibility in a future article.

Next, the InstallerSection class comes with ten methods, some of which you can override. This article, however, will focus only on those methods that a basic plug-in can use. For instance, to get the current panel, use the `firstPane` or `activePane` method. Either method will return the same InstallerPane instance if the plug-in has only one custom panel.

```
tPnl = [[self section] firstPane];
```

To get the nib that carries the custom panel, send a `bundle` message to InstallerSection. This returns the bundle as an NSBundle object.

```
tBndl = [[self section] bundle];
```

To read the title string for the active panel, call the `title` method. InstallerSection responds by returning the panel title as an NSString.

```
tTitle = [[self section] title];
```

And to find out the current install state, use the `state` accessor. This gives you an instance of the InstallerState class, which is described next.

```
tState = [[self section] state];
```

## The InstallerState class

As stated earlier, the InstallerState class (Figure 7) returns the current state of the install session. Like InstallerSection, this class is

# Don't ruin your client's web presence with dull typography.

Transform this


Into this




1. The main title <H1> has been converted to a photofont that is consistent with the desired look and feel of the site. With photofonts designers can take advantage of gradients, transparency, colors, shadows and more!
2. Until now, web designers had to settle for Arial, Georgia or some other "safe" font to make sure the overall design of the page looked consistent. With **Photofont WebReady** designers can use the most suitable font for each project.

**Photofont WebReady** allows you to enhance your web pages with fonts of your own choice in a search engine friendly, standards-compliant way. With **Photofont WebReady**, you can convert any photofont, OpenType font or TrueType font into an embedded web font. The web font is then rendered on your web page using Flash® technology, yet keeping all the advantages of standard hypertext. Your visitors see the page the way you want them to see it, and search engines see it the way you want *them* to se it.

Learn more about **Photofont WebReady** and photofonts at **http://www.photofont.com/photofont/webready/**

Photofont
WebReady
A FONTLAB PRODUCT

instantiated by InstallerPane once the latter displays its panel. To access the instance, use the `state` accessor of InstallerSection.

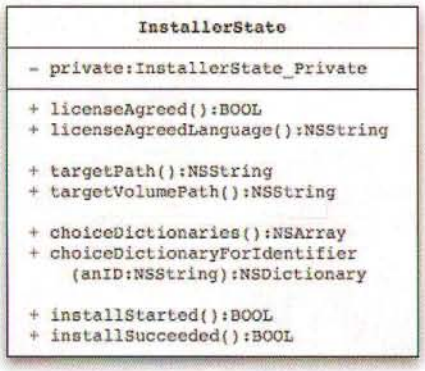



**Figure 7. The InstallerState class**

There are four sets of methods in the InstallerState class. Each set corresponds to a specific stage in the install session. The first set returns the results of the **License** panel. To find out if users accepted the license terms, use the `licenseAgreed` method. The method returns a `YES` if users did accept the terms; otherwise, it returns a `NO`.

```
tAgree = [[[self section] state] licenseAgreed];
```

To find out which localized license is displayed, use `licenseAgreedLanguage`. This method returns the language as an NSString.

```
tAgree = [[[self section] state] licenseAgreedLanguage];
```

The second set of methods return the results of the **Select Destination** panel. They tell the plug-in where the package will install its payloads. To get the selected target volume, use the `targetVolumePath` method.

```
tVol = [[[self section] state] targetVolumePath];
```

The method returns the mount point of the selected volume as an NSString. To get the final destination for the payload, call the `targetPath` method.

```
tPth = [[[self section] state] targetPath];
```

This returns the destination's absolute path as an NSString. The returned path will also have the volume's mount point as part of its string.

The next set of methods give the results of the **Custom Install** panel. They tell the plug-in which payloads where chosen by the users. For a list of all payloads, call the method `choiceDictionaries`.

```
tList  = [[[self section] state] choiceDictionaries];
```

The method returns its list as an NSArray. For a specific payload choice, use the method `choiceDictionaryForIdentifier`. Then pass the payload's choice ID as input.

```
tPayload  = [[[self section] state]
        choiceDictionaryForIdentifier:@"foobar"];
```

This returns the choice settings as an NSDictionary. There are three entries in this dictionary, each entry with its own unique key. The choice ID, for instance, uses `InstallerState_Choice_Identifier` as its key, the payload's destination path `InstallerState_Choice_CustomLocation`. And the choice state is under the key `InstallerState_Choice_Installed`. Figure 8 shows which field on the choice's **Configuration** panel corresponds to which key.

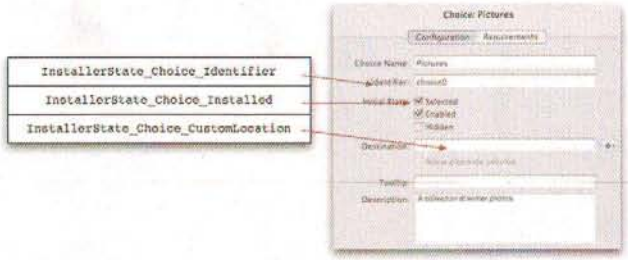



**Figure 8. The choice states and their keys**

# The Plug-in Template

To create an installer plug-in, use the Xcode project template, aptly named, **Installer Plugin**. Xcode 3.0 files this template in the directory `/Developer/Library/Xcode/Project Templates/Standard Apple Plug-ins`. You can, of course, write your own plug-in project from scratch. Using the template, however, reduces the amount of guesswork on your part.

Figure 9 shows the bundles and files of the project template. Note that there are two project bundles: one with a `.xcode` suffix, the other `.xcodeproj`. Use the `.xcodeproj` bundle if your Xcode IDE is version 2.x or newer; use the `.xcode` bundle for older versions of Xcode.

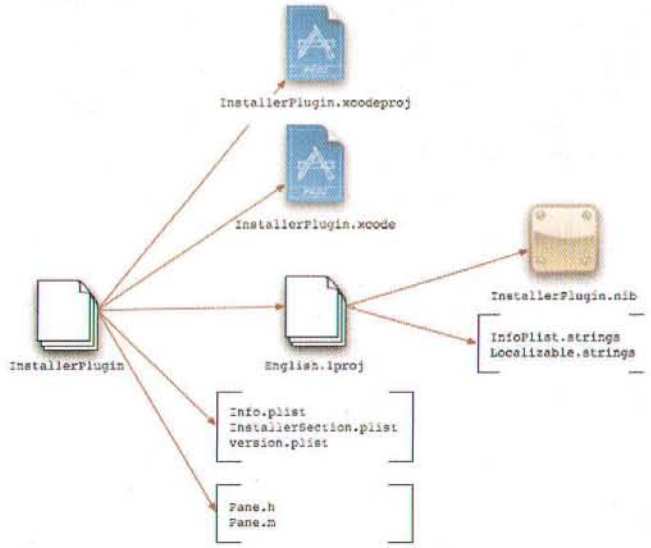
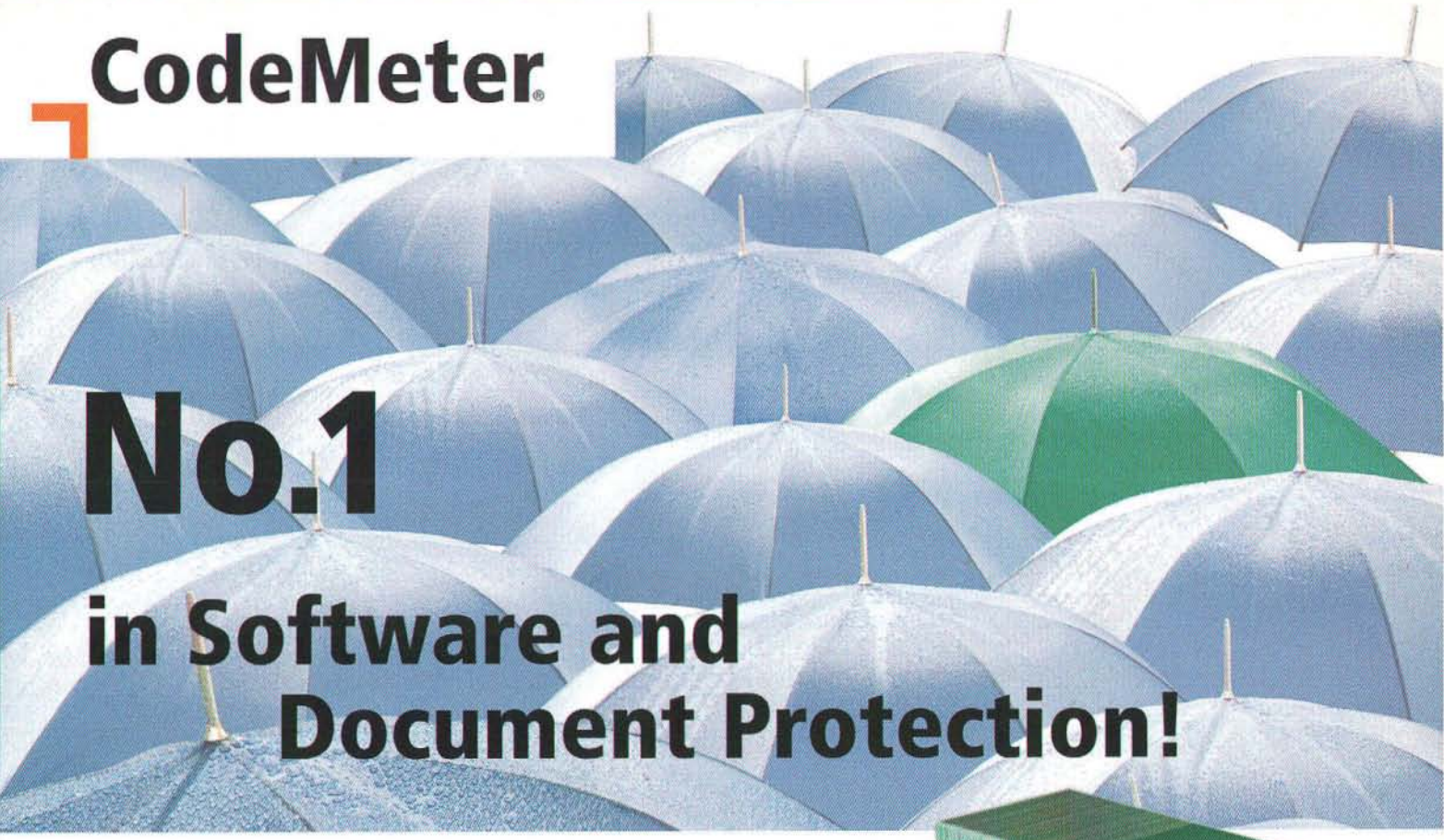



**Figure 9. The plug-in project template**

Note also that there are only three items that you should be updating. The rest of the project items have default settings that will suffice for most cases.

# CodeMeter®



# No.1
## in Software and Document Protection!

### Highest Security
- Vendor selectable secret and private key.
- Strong encryption algorithms with AES 128-bit and ECC 224-bit.
- Best-in-class tools for automatic protection (envelope, without source modification) for Win32, Win64, .NET, Java and MacOS X Universal (PPC, Intel).

### Best Flexibility
- More than 1000 independent licenses can be protected by one CM-Stick.
- One versatile hardware key for all license models including floating network licenses.
- Multi platform support including Windows, MacOS X and Linux.

### New Distribution Channels
- License transfer by SOAP based CM-Talk or file based Field Activation Service in e-shops.
- Multiple-purpose, including protecting low cost software and digital content.

### Unique End User Advantages
- First and smallest dongle with up to 2 Gbyte flash drive.
- No drivers necessary – can be used without administrator rights.
- CM Password Manager, secure virtual drive and secure login.


**Order your Free Software Development Kit now!**
**Phone 1-800-6-GO-WIBU | order@wibu.us**

2007 SIIA //CODiE// FINALIST
2005 SIIA //CODiE// FINALIST
2006 SIIA //CODiE// FINALIST

Mac

Universal

WIBU-SYSTEMS submitted the CodeMeter Password Manager and the CodeMeter SDK for the **Apple Design Awards 2007.**

WIBU SYSTEMS

**WIBU®-SYSTEMS USA Inc.**
110 W Dayton Street
Edmonds, WA 98020, USA
www.wibu.com
info@wibu.us
Tel: 1.800.6.GO.WIBU
1.425.775.6900
Fax: 1.206.237.2644

## The InstallerSection.plist file

This file defines where your custom panel will appear in the install session. This file must be placed in the `Contents/Plugins` directory of your installer package. Without this file, your package will ignore your installer plug-in.

Listing 1 shows the default contents of that file. Note the file list only six of the panels that appear in the install session. The first three on the list refer to the Welcome, Readme, and License panels. The `Target` entry is the Select Destination panel, and `PackageSelection` is either the Custom or Standard Install panel. Finally, the Install entry is the progress panel, which appears when the package starts installing its payloads.

### Listing 1. The InstallerSection.plist file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST
1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>SectionOrder</key>
    <array>
        <string>Introduction</string>
        <string>ReadMe</string>
        <string>License</string>
        <string>«PROJECTNAME».bundle</string>
        <string>Target</string>
        <string>PackageSelection</string>
        <string>Install</string>
    </array>
</dict>
</plist>
```

Note as well that the file places the plug-in between the `License` and `Target` entries. It also lists the plug-in under the generic name of `PROJECTNAME`. When you create your plug-in project using this template, Xcode replaces `PROJECTNAME` with your plug-in's name. If you move your plug-in's entry in the list, you change the position of its panel in the install session. Also, if you delete one of the entries in the list, you prevent that panel from appearing during the session. But be careful with this last step, as it may lead to unexpected results.

## The InstallerPlugin.nib bundle

This bundle defines the look and feel of your plug-in's panel. Like all nib bundles, you use the Interface Builder to make changes to this bundle. Figure 10 shows the five default objects in this bundle. Two of the objects are for your plug-in's use. For instance, the View object is, obviously, an instance of NSView. This object carries the interface of your custom panel. Double-clicking it gives you an empty panel that is 418 pixels by 330 pixels in size. Do not, however, change the size of the panel. If you do, your interface widgets will appear misaligned during the install session.

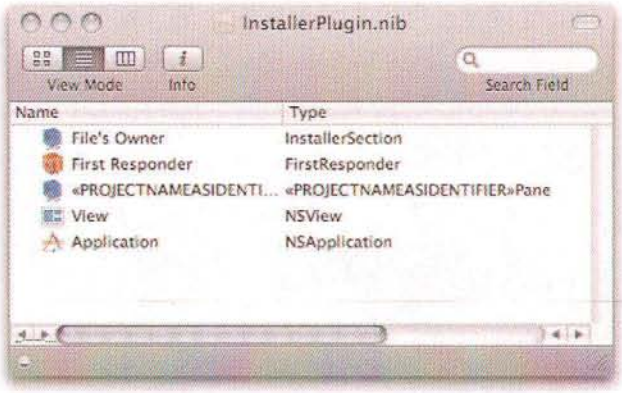



**Figure 10. The InstallerPlugin.nib bundle**

The <<PROJECTNAMEASIDENTIFIER>>Pane is an instance of InstallerPane. Its name is set to the project's name after you create the project. For instance, if your plug-in project is `Foobar`, the InstallerPane object gets the name `FoobarPane`.

The other three objects are the usual proxy objects you find in most nib bundles. But in this nib, the File's Owner proxy refer to the instance of InstallerSection. Its one outlet, `firstPane`, is linked to the nib's InstallerPane instance. The Application proxy refers to the global NSApplication object, which will be the Installer utility for the plug-in. And the First Responder proxy refer to the first object in the responder chain. This proxy is currently not attached to any object on the nib.

## The Pane files

Finally, there are the two files: `Pane.h` and `Pane.m`. These files define the InstallerPane controller for your panel. Again, when Xcode creates your plug-in project, it prefixes the project name to each file. Do not, however, change the assigned file names to something else. If you do, your plug-in will be unable to display its panel correctly.

Treat these files as you would any controller class files. For instance, you can add outlets and link them to specific widgets on the custom panel. You can define actions that your panel widgets can call at runtime. You can even add code to manipulate user data or pass that same data to your own custom model class.

## CAVEAT

At the time of writing, the plug-in template has one interesting flaw. When you use it to create a project, the project's nib bundle loses all six outlets for its InstallerPane instance. Your plug-in will still work with this flaw present, but it will be unable to query the current install state.

To work around this flaw, save your nib bundle as `NIB 2.x`. With a text editor, open the file `classes.nib` that is inside your bundle. Then add the lines in Listing 2 to that file. These lines will restore the six outlets to your bundle. Save your changes when done.

Whether this flaw is fixed in Xcode 3.1 is unconfirmed.

**Listing 2. Adding outlets to classes.nib**

```
<key>OUTLETS</key>
<dict>
    <key>contentView</key>
    <string>NSView</string>
    <key>firstKeyView</key>
    <string>NSView</string>
    <key>initialKeyView</key>
    <string>NSView</string>
    <key>lastKeyView</key>
    <string>NSView</string>
    <key>nextPane</key>
    <string>InstallerPane</string>
    <key>parentSection</key>
    <string>id</string>
</dict>
```

# To Create A Plug-in

Let us now build a simple plug-in using the Installer Plug-in template. Our plug-in will ask users to enter their name, company, and product serial number during the install session. If the serial number is correct, the plug-in will enable the Continue button. Otherwise, it will display an alert dialog and disable the same button.

So, start up your copy of Xcode. Choose New Project from the File menu and pick Installer Plug-in from the list of project templates. Click the Continue button and set the project's name to `Register`. Leave the project's directory at the default location. Click the Finish button to create and open your new project.

## Defining the panel

Select the entry `RegisterPane.h` from the Groups & Files pane. Add the following outlets to that file's `@interface` block.

```
IBOutlet NSTextField    *oUser;
IBOutlet NSTextField    *oComp;
IBOutlet NSTextField    *oSerial;
```

Then add the following action to the same block.

```
- (IBAction) registerCheck:(id)aSnd;
```

Next, double-click the `Register.nib` entry from the pane, thus opening the bundle in Interface Builder. From the Register window, select the icon «PROJECTNAMEASIDENTIFIER»Pane. Choose Identity Inspector from the Tools menu and set the Class field to `RegisterPane`. Scroll down to the Interface Builder Identity pane and set the Name field to `RegisterPane`. Save your changes by choosing Save from the File menu. To find out if your changes are correct, go to the Class Outlets pane of the palette. You should see your three outlets listed in that pane. Close the palette when you are done.

Now double-click the View icon on the Register window. Lay out the panel as shown in Figure 11. There are two sets of widgets on the panel. The first set consists of NSTextFields serving as static labels. The second set consists of NSTextFields serving as editable fields.

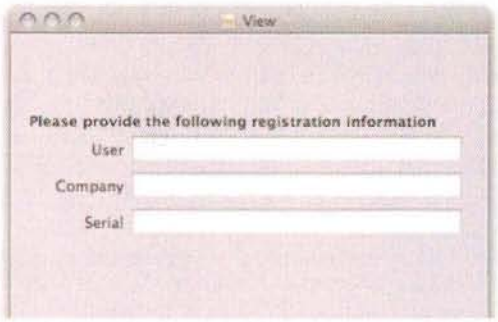



**Figure 11. Layout of the custom panel**

Select the RegisterPane icon and choose Connections Inspector from the Tools menu. With your pointing device, link each editable field to the right outlet (Figure 12, *red*). For this plug-in, the `oUser` outlet links to the User field, the `oComp` to the Company field, and `oSerial` to Serial. Then link all three editable fields to the `registerCheck` action (*blue*). Take care to leave the template's preset links alone. Save your changes and switch back to the Xcode editor.

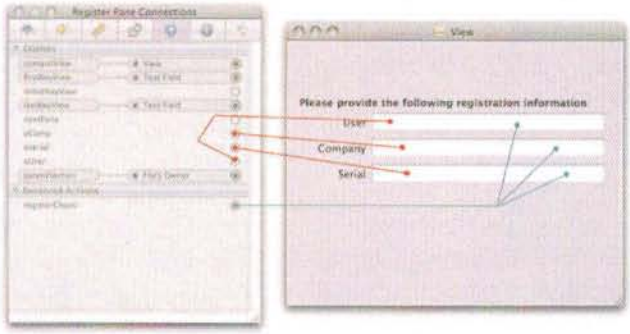



**Figure 12. Linking the panel widgets**

## Implementing the panel

We now enter the code-writing part of the project. Select the entry `RegisterPane.m` from the Groups & Files pane. Go to the delegate method `didEnterPane` and enter the code shown in Listing 3. This method sets each of the outlets to their default values. Then it disables the Continue button and enables the Go Back button.

**Listing 3. Setting the default panel data**

```
- (void)didEnterPane:(InstallerSectionDirection)aDir
{
    // initialize the following outlet fields
    [oUser setStringValue:@"your name goes here"];
    [oComp setStringValue:@"your company name"];
    [oSerial setStringValue:@"123456789"];

    // disable the Continue button
    [self setNextEnabled:NO];

    // enable the Go Back button
    [self setPreviousEnabled:YES];
}
```

Next, enter the code in Listing 4 to the delegate method

shouldExitPane. First, the method reads the hash values from the oUser and oComp outlets. It then combines the two hash values and compares the results with the value from oSerial. If both values are different, the method displays a warning dialog to the users. It then disables the Continue button and returns a NO to prevent the panel change. On the other hand, if both values are the same, the method returns a YES to allow the change.

## Listing 4. Checking the user data

```
- (BOOL)shouldExitPane:(InstallerSectionDirection)aDir
{
    NSUInteger tUsr, tCmp, tXor, tSN;
    NSAlert *tWrn;

        // check the direction of movement
    if (aDir == InstallerDirectionForward)
    {
            // read the hash values of each registration
        tUsr = [[oUser stringValue] hash];
        tCmp = [[oComp stringValue] hash];
        tXor = tUsr ^ tCmp;

            // read the serial number
        tSN = [oSerial intValue];
        if (tSN != tXor)
        {
                // create a warning dialog
            tWrn = [[NSAlert alloc] init];
            if (tWrn != nil)
            {
        // initialize the dialog
                [tWrn addButtonWithTitle:@"OK"];
                [tWrn setMessageText:@"Invalid serial
number"];
                [tWrn setInformativeText:
              @"Please check and re-enter your registration
information."];
                [tWrn
setAlertStyle:NSInformationalAlertStyle];

                    // display the warning dialog
                [tWrn runModal];

                    // dispose the warning dialog
                [tWrn release];
            }
                // disable the Continue button
            [self setNextEnabled:NO];

              // prevent the panel movement
            return (NO);
        }
    }
    // allow the panel movement
    return (YES);
}
```

Finally, to the registerCheck action, enter the code in Listing 5. This action checks the data in each outlet. If all outlets have a non-zero length string, the action then enables the Continue button. Otherwise, that same button remains disabled. Save your changes after you have updated these three methods.

## Listing 5. Responding to the user entry

```
- (IBAction) registerCheck:(id)aSnd
{
    BOOL tChk;

      // check the registration fields
    tChk = ([[oUser stringValue] length] > 0);
    tChk &= ([[oComp stringValue] length] > 0);
    tChk &= ([[oSerial stringValue] length] > 0);

    // enable the Continue button
    [self setNextEnabled:tChk];
}
```

Now, a few words before we proceed to the next part of our project. First, note the use of the NSString hash function to generate the values for oUser and oComp. While this function is fine for our sample plug-in, it is impractical for real-world use. Future versions of NSString may behave differently. As a result, their hash functions may return a different value for the same string. A more reliable solution is for you to use your own hash algorithm.

Second, the plug-in only checks if the registration data is valid before it allows or deny product installation. This, again, is impractical because users can defeat the check by removing the plug-in. One good solution is to have the plug-in write its results to a hidden file. The installed payload can then look for this file and even read its data. If the data is correct, the payload behaves normally. If not, or if the file missing, the payload displays a reminder dialog or it runs in demo mode.

## Building the plug-in

We are now ready to build our plug-in. But first, we must define where our plug-in's panel will appear in the install session. In this case, we want our panel to come after the Select Destination panel. Select the entry InstallerSection.plist from the Groups & Files pane. Locate the entry Register.bundle and move it after the entry for Target (Listing 6). Save your changs when done.

## Listing 6. Modified contents of InstallerSection.plist

```
    <array>
        <string>Introduction</string>
        <string>ReadMe</string>
        <string>License</string>
        <string>Target</string>
        <string>Register.bundle</string>
        <string>PackageSelection</string>
        <string>Install</string>
    </array>
```

Now build the plug-in by clicking the Build button on the Xcode toolbar. Xcode compiles each project file and links them to create the plug-in bundle. It then places the bundle, named Register.bundle, in the in the project subdirectory build/Release/.

## Installing and testing the plug-in

To test the plug-in, you will need a basic installer package. Now when you prepare your installer project, make sure to set its Minimum Target to MacOS X 10.4. This will tell PackageMaker to use the distribution bundle as the package format.

In this article, we will use Foobar_Demo as our installer project. Its payload consists of three TIFF files, which will go into the directory /Users/Pictures. Build the package by choosing Build from the Project menu. When prompted, use Foobar as the package name. Go to the Finder and control-click the package to display its contextual menu. Choose Show Package Contents to open the bundle in a separate Finder window. Go to the

Contents directory and create a new subdirectory named Plugins. Then copy the plug-in Register.bundle and the file InstallerSection.plist into that subdirectory (Figure 13). Close the Finder window when done.

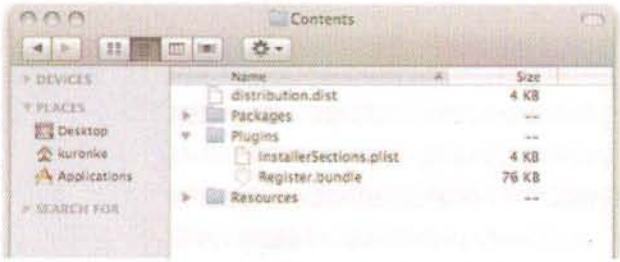



**Figure 13. Installing the plug-in and plist**

Now double-click the Foobar package to start the install session. First, you get a modal dialog (Figure 14) warning you that the package will perform a custom task. This means the package recognizes your plug-in's presence. But it can also mean that the package contains an install action or script. Click Continue to proceed.

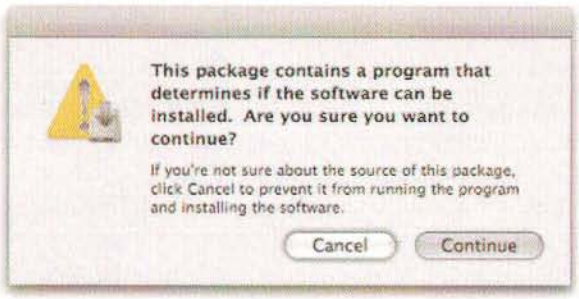



**Figure 14. Warning the user**

After the package displays its Welcome panel, you will see the name Register added to the list of panels (Figure 15, *orange*). You will also see that same name after Destination Select. This means the package knows that the plug-in has a custom panel, and it knows when to display the panel.


**Figure 15. The Welcome panel with an updated list**

Click the Continue button until you reach the Register panel. For the User field, enter the name "Alan Smithee" as the user. For the Company field, enter "Foobar". Leave the Serial field unchanged. You should see the Continue button enabled at this point. Now click that button. The package will display a warning dialog (Figure 16) telling you that the registration data is wrong.

Dismiss the dialog by clicking its OK button. Carefully type 1239302760 into the Serial field and then click the Continue button. This time, the package will display the next panel, which is the Custom Install panel.

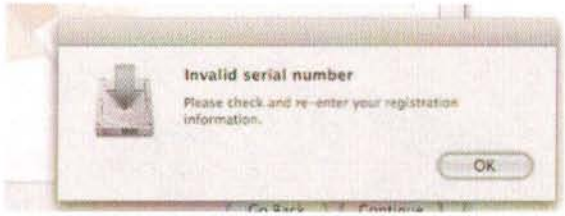



**Figure 16. Incorrect registration**

## Concluding Remarks

An installer plug-in is another way to customize your install session. With a plug-in, you can display a custom panel for other users to interact. You can use the Cocoa framework to do tasks that are hard, if not impossible, to do with an action or script.

Xcode comes with a basic template that you can base your plug-in project. This template sets the necessary files and nibs needed for such project. Adding a plug-in is as simple as a drag and drop. But keep in mind that only a meta-package and a distribution package will use plug-ins. The new flat-file package, introduced in 10.5, will ignore any plug-ins it may carry.

## Recommended References

As stated earlier, Apple has yet to document the task of writing an installer plug-in. So far, your best options, besides this article, are to study their sample plug-in project, or the header files of the InstallerPlugin framework. You can also query the Installer-dev list archives for possible answer to your questions.

Stéphane Sudre, maker of Iceberg, also wrote a couple of pieces on the topic. Listed below are his online works for your benefit.

Stéphane Sudre. "Defining Installer Plugins". *Iceberg Users Guide*. Copyright 2008. Accessed on 2008 Aug 14. Online:
http://s.sudre.free.fr/Software/Iceberg.html

Stéphane Sudre. "Installer Plugins". *Installation – The Lost Scrolls*. 2008 Apr 10. Accessed on 2008 Au8r 14. Online:
http://s.sudre.free.fr/Stuff/Installer/Installer_Plugins/index.html

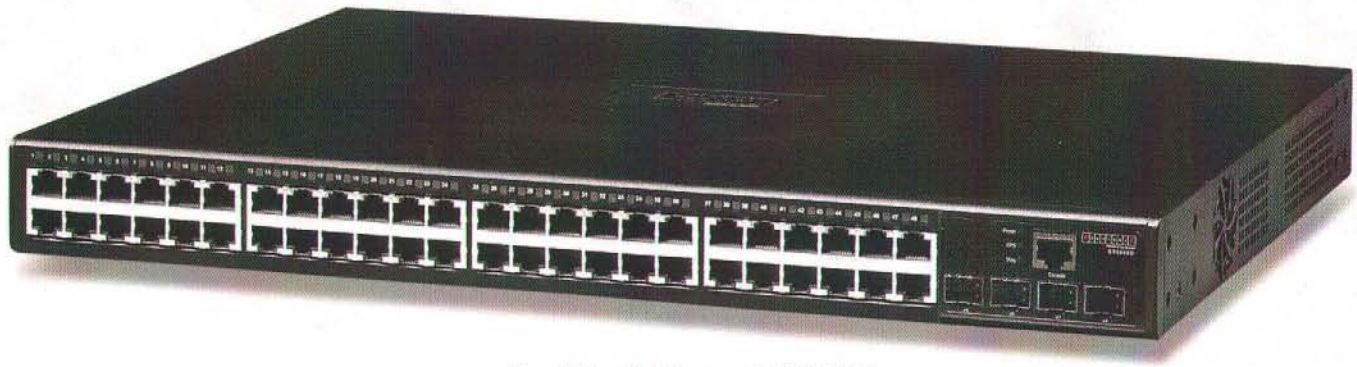
## About The Author

***JC is a freelance engineering writer from North Vancouver, British Columbia. He spends his time writing technical articles; tinkering with Cocoa, REALbasic, and Python; and visiting his foster nephew. He can be reached at anarakisware@gmail.com.***

# Small Tree's Edge-corE ES4548D

## Cost effective, high-bandwidth switch for your network

*By Dennis Sellers and Neil Ticktin*



**Small Tree's Edge-corE ES4548D**

## The Need

Everyone talks about wireless, but when it comes to fast, reliable, trouble free connectivity, a wired Ethernet connection is still king. All of Apple's machines come with gigabit Ethernet, but that's only part of the solution. Sure, there are cheap 5-port switches, but what if you need 24- or 48-ports? Gigabit for all those ports? It'll cost you an arm and a leg, right? Not necessarily.

MacTech has been using a solution in house, and put it through a variety of real world scenarios and every day use for a sustained period of time.

## Enter Small Tree's Edge-corE ES4548D

If you need a L2 Gigabit Ethernet switch for bandwidth-intensive networks you might check out Small Tree's Edge-corE ES4548D. It's designed for networks that need advanced switching features such as line-rate rate performance, dynamic 802.3ad link aggregation, IPv6 support, security, high availability and advance QoS to the network edge while offering the simplicity of traditional LAN switching.

The Edge-Core ES4548D is designed for high performance server aggregations, such as enterprise data centers, to connect high-end or network attached file servers over copper ports. High-speed workgroups backbone upgrades, and Gigabit to the desktop for power users. The whole stack can be managed as a single entity with a single IP address.

The ES4548D sports 44 x 10/100/1000Base-T + 4 Gigabit Combo (RJ45/SFP) ports with Gigabit Ethernet link on all ports. Its advanced QoS features are for "triple play" performance, while the switch is very scalable. It has a forwarding rate of 71.5Mpps, a MAC Address Table size of 8K and a switching capacity of 96Gbps. With its 96Gbps switching capacity, the ES4548D delivers wire-speed switching performance on all gigabit ports. There are four Gigabit Ethernet combo ports for uplink flexibility, allowing copper or fiber uplinks.

IEEE 802.1w Rapid Spanning Tree Protocol provides a loop-free network and redundant links to the core network with rapid convergence. IEEE 802.1s Multiple Spanning Tree Protocol runs STP per VLAN base, providing Layer 2 load sharing on redundant links. IEEE 802.3ad (LACP) is designed to increase bandwidth by automatically aggregating several physical links together as a logical trunk and providing load balancing and fault tolerance for uplink connections.

IGMP snooping prevents flooding of IP multicast traffic and limits bandwidth intensive video traffic to only the subscribers.

Comprehensive QoS support with 8 egress queues per port enable differentiated management of up to eight traffic types. Traffic is prioritized according to 802.1p, DSCP, IP precedence and TCP/UDP port number; the goal is to give optimal performance to real-time applications such as voice and video. Asymmetric bidirectional rate-limiting, per port or per traffic class, preserves network bandwidth and allows maximum control of network resources.

IEEE 802.1Q-in-Q allows the service provider to provide certain services, such as Internet access on specific VLANs for

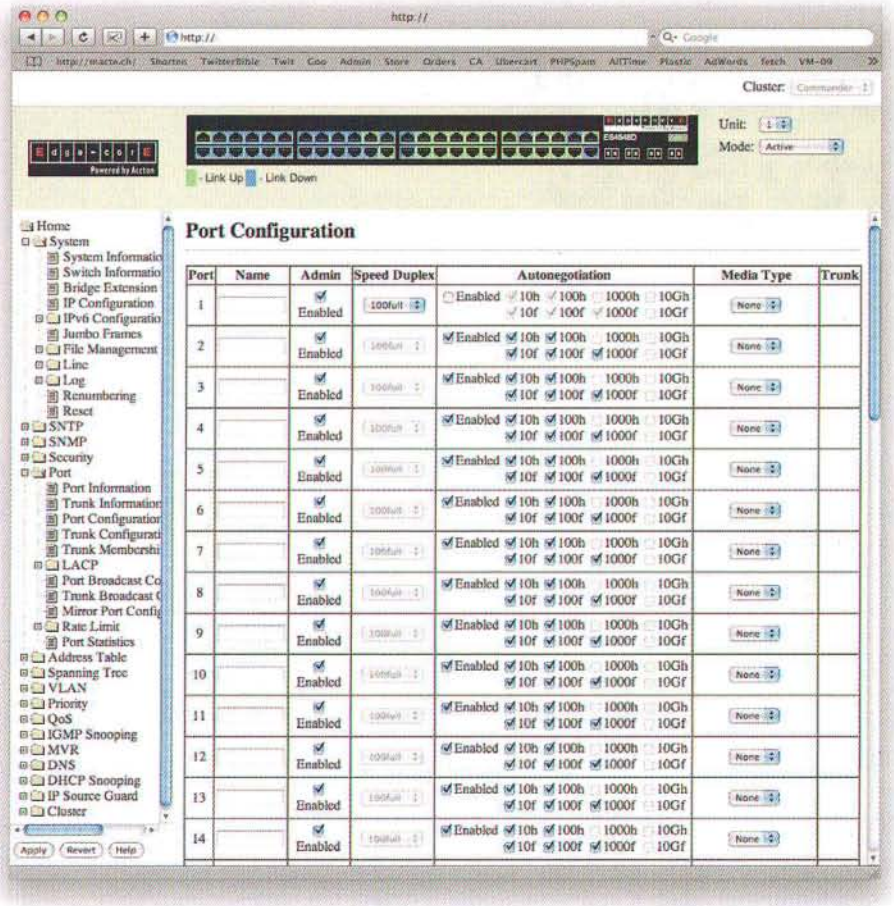
specific customers. It provides other types of services for their other customers on other VLANs.

Port Security ensures access to a switch port based on MAC address. IEEE 802.1X port-based or MAC-based access control makes sure that all users are authorized before being granted access to the network. User authentication is implemented via any standard-based RADIUS server. Access Control Lists (ACLs) can be used to restrict access to sensitive network resources by denying packets based on source and destination MAC addresses, IP addresses, TCP/UDP ports.

SSL, Web Management Encryption, RADIUS and TACACS+ are designed to protect data communication and ensure data privacy. What's more, Private VLAN isolates edge ports to ensure user privacy.

You configure the ES4548D by using the web-based graphical user interface. Industry standard Command Line Interface (CLI) via console port, Telnet or SSH provides a common user interface and command set for users to manipulate the switch.

Four groups of RMON are supported. You can back-up and restore firmware and configuration files via TFTP.




**Web Interface**

The ES4548D supports IPv6 management, QoS and security to help you get ready for future transition to IPv6. IPv6 can support an exponentially greater number of IP addresses compared to IPv4. The internals of the IPv6 protocol have been designed for scalability and extensibility. This means that a variety of different kinds of devices besides computers, such as cell phones and home appliances, will be able to more easily join the Internet in future, according to the folks at Small Tree.

The ES4548D lists for $2,024. Also, Small Tree offers an optional redundant power supply for the ES4548D that provides uninterrupted power.

http://www.smalltree.com/Edge_corE_48_port_L2_Gigabit_Standalone_Switch_p/es4548d.htm

## But, Is It Worth It?

All the details are nice, but the end result is what is really important. Is it worth upgrading 100 Mbps switches to Gigabit?

Not every application can take advantage of the faster bandwidth, but in our testing, some of the most burdensome network issues (such as backups and file copies) really fly. In real world scenarios, we continually saw a 4-6x speed improvement over what 100 Mbps delivered under the same scenarios. Even interaction with the email servers when sending enclosures was noticeably faster. We often saw backups run, even with slower machines as clients, at nearly 1 gigabyte per minute (i.e., approximately 480 Mbps).

So, is it worth it? Definitely. At least that's what we found with Small Tree's ES4548D. You get the performance of some of the "name" brands, for a reasonable price.

MT

## About The Authors

***Neil Ticktin is the Editor-in-Chief and Publisher of MacTech Magazine. Neil has been in the Mac industry since 1985, has developed software, written documentation, and been heading up the magazine since 1992. When Neil does a benchmark article, he likes to test the features that people will use in real-life scenario and then write about that experience from the user point of view. Drop him a line at publisher@mactech.com***

***Dennis Sellers is a long time journalist. He started in the newspaper business, but has been in the online journalism business for the past 15 years. He's the editor/publisher of Macsimum News (http://www.macsimumnews.com)***

# Etymotic etyBLU Headset

*By Dennis Sellers*

The folks at Etymotic say their etyBLU (http://www.etymotic.com/ephp/etyblu.aspx) was the first Bluetooth headset that features a connecting noise-canceling boom microphone. Described as a "dual mode" Bluetooth headset, it comes with a noise canceling boom microphone (the Blumaxx Quick-Connect Microphone, which enables callers to be heard clearly and understood) and an internal microphone. You can use the latter independently simply by unplugging the boom microphone. As for the BluMaxx, it gives you up to 25 decibels more noise cancellation compared to the internal microphone.

When it's connected correctly, the opening at the tip of the microphone is closest to the mouth and the opening at the middle of the microphone tip faces away from the mouth. You can connect or disconnect the Blumaxx without disconnecting a call.

At US$129, the etyBLU isn't inexpensive. Of course, you're paying for two microphones. Why two? According to Etymotic, the combination of an in-ear noise-isolating earphone and a boom mic provides better sound quality and clearer communication at both ends of a conversation. That said, the headset is designed for taking and making phone calls, not for playing your tunes.

The etyBLU weighs less than .5 ounces and has a range of around 30 feet. It boasts up to seven hours of talk time and up to 100 hours of standby time. It takes 2-3 hours to fully charge the headset; you can get an 80 percent charge after one hour.

Unlike Apple's iPhone Bluetooth Headset, which has a foam-covered earbud, the etyBLU uses an in-canal earpiece. It comes with two sets of small and regular-sized triple flanges, and one foam alternativeNote whose ear tip fits snugly in the ear. So snugly that some people find it uncomfortable. But it's great at blocking out noise.

In fact, it's so good at it that you should take the etyBLU off when you're not using it so you can hear the world around you. For that reason, be very careful when you're using it while driving (and many folks will use it for just that to allow hands-free conversation) as you won't be able to hear through one ear.

The earpiece in the etyBLU headset has a special filter that smoothes the frequency response and prevents earwax from entering the earphone. The filter is located at the end of the earphone and is visible when the ear-tip is removed. You should change the filter if loudness decreases or the sound quality declines. Filters aren't reusable, but the etyBLU comes with an extra earpiece filter and filter replacement tool.

There are two volume buttons located on the side of the etyBLU and one port on its bottom. The port is where you can plug in an included USB charging cable or the boom mic.

Etymotic also tosses in a metal ear-hook, which can be used for extra stability, and a foam windscreen.

The etyBLU has a slim black design with a silver multifunction button that calls to mind (probably intentionally) the Apple Bluetooth Headset. It measures 1.8 inches long by 0.5 inches wide by 0.5 inches thick. There's also an LED indicator light underneath the button.

The etyBLU offers features you'd expect in such a headset: the ability to answer, end, and reject calls, last number redial, voice command support, call waiting support, three-way calling support, and the ability to transfer a call to the handset and vice versa.

When you're ready to use the etyBLU headset, it's discoverable for five minutes after you power it on. To pair it with a device, such as an iPhone, you:

- Turn the phone OFF and then back ON.
- Turn on the mobile phone's Bluetooth function.
- Press and hold the Multi-Function Button (MFB) for five seconds. The blue LED will flash rapidly when ready for pairing.The etyBLU stays in pairing mode for five minutes.
- Activate the search for BT and/or audio devices on the phone When the etyBLU is found, the phone will display "etyBLU." Scroll to it and press OK to confirm pairing.
- If a passkey/password is requested, enter 0000.

The etyBLU is compliant with Bluetooth Core Specification 2.1, Headset 1.1 and Handsfree 1.5 profiles. It comes with a two-year warranty.

MT

## About The Author

***Dennis Sellers is a long time journalist. He started in the newspaper business, but has been in the online journalism business for the past 15 years. He's the editor/publisher of Macsimum News (http://www.macsimumnews.com)***

# Advertiser/Product Index

## By Company



## By Product

# Philip Goward and Greg Scown

## SmileOnMyMac

http://www.smileonmymac.com/

**What do you do?**

Company Co-Founders

**How long have you been doing what you do?**

Greg Scown: I've been an indie software developer for over seven years, since 2002. My first job out of college was with Apple. I had a few jobs between then and now, mostly oriented toward raising money to take another shot at being an indie software developer.

Philip Goward: I've been with SmileOnMyMac for close to 6 years now. Otherwise I've been working in software for about 20 years, with a focus on objects, user interface and graphics.

**What was your first computer?**

Greg Scown: TRS-80 Color Computer from Radio Shack. My first Mac was a PowerBook 140.

Philip Goward: Sinclair ZX-81 until I saved for an Acorn Atom.

**Are you Mac-only, or a multi-platform person?**

Greg Scown: I'm a Mac person, but not quite a Mac zealot. I have some familiarity with the forces of the dark side.

Philip Goward: I've worked on many systems, but the Mac is where my heart is.

**What attracts you to working on the Mac?**

Greg Scown: Community. There's an excellent community of users, developers, and fans of the Mac. They're demanding, and they're great.

Philip Goward: Quality. Folks developing for the Mac love what they do, and it shows.

**What's the coolest thing about the Mac?**

Greg Scown: The whole package. The fact that the sum of the hardware and software is greater than either alone.

Philip Goward: With the Mac the beauty is not just skin deep. The software on a Mac is as well-designed as machines themselves. Even my technology-averse Mother can use it.

**What is the advice you'd give to someone trying to get into this line of work today?**

Greg Scown: Follow your passion. Engage the Mac community. Work really hard.

Philip Goward: Focus on what you are learning each step along the way.

**What's the coolest tech thing you've done using OS X?**

Greg Scown: I wrote code to flow text around a circle live while typing. This taught me that Mac OS X was the real deal, and it was part of the inspiration for DiscLabel.

Philip Goward: I'm partly stumped because I find the cool factor to always be in algorithms that use the OS routines to their max. Otherwise, the Quartz Composer work in BrowseBack that makes the screen slide diagonally backwards is pretty cool.

**Where can we see a sample of your work?**

Everything we do at SmileOnMyMac is collaboration. You can look at our work at: http://www.smileonmymac.com/

**The next way I'm going to impact IT/OS X/the Mac universe is:**

Greg Scown: Listen to our customers. Continue to improve our software.

Philip Goward: ...To keep on going.

**Anything else we should know?**

Greg Scown: I got started on the Mac because I helped answer a PC question in my college library. I happened to do so in front of the lab manager, and she offered me a job. We had 70 Macs and 6 PCs. I learned an awful lot about Macs in a short time at that job.

Philip Goward: I've never looked back since my SE30.

***If you or someone you know belongs in the MacTech Spotlight, let us know! Send details to editorial@mactech.com***

FIND THE PERFECT GIFT.

# DADS & GRADS

Smalldog.com/fathersday Smalldog.com/graduation

**Small Dog Electronics has over 3,000 potential gifts in stock, so you're sure to find that perfect something. Plus, we offer:**

» Free shipping on purchases over $200
» Mac bundles and tax-free shopping

## FEATURED GIFT IDEAS

*For Dad, recent grad or anyone!*

» Purchase **any Mac with AppleCare** and get a **free Canon MP190 all-in-one*** 
» Small Dog exclusive rebate; visit **Smalldog.com/sderebate** for details
*After mail-in rebate


*Everybody loves iPods*

» 16GB iPod touch for the 8GB price – **$229.99**
» iPod specials from $159.99

*Perfect for the college dorm*

» Buy the **iPod Essentials 4-Pack (Chill Pill + Adapters + RapCap mic)** for just **$79.99** (save $40)

**Small Dog Electronics**

*Always By Your Side*

- over 15 years in the Mac community
- 3,000+ products for Macs + PCs
- 5-star online merchant rating
- tax-free shopping outside of VT

**www.smalldog.com** **800-511-MACS** Apple Specialist

# Amp Up Your Mac!™

## Call MacMall for our latest specials with your NEW Mac!


**6 Months Same as Cash!**
*Valid for purchases over $500. Call for details.*

**Up to $250 Cash Back!***
*On select Apple computers from our Web site at www.macmall.com. After mail-in rebate.*

**FREE Parallels Desktop!***
*After mail-in rebate with purchase of an Apple computer.*

### NEW 15" Aluminum MacBook Pro
2.4GHz with 2GB SDRAM and 250GB Hard Drive
FREE Parallels Desktop!* *Includes NEW iLife '09!*

$1994 - $150 mail-in rebate* = **$1844!** #7684020
*After mail-in rebate.


**SAVE $150!**
17" MacBook™ Pro
2.66GHz with 4GB SDRAM, 320GB HD and SuperDrive
FREE Parallels Desktop!*
$2794 - $150= **$2644!*** #7732924 *After mail-in rebate.


**SAVE $50!**
NEW 20" iMac®
2.66GHz with 2GB SDRAM, 320GB HD and SuperDrive
FREE Parallels Desktop!*
$1194 - $50 = **$1144!*** #7758229 *After mail-in rebate.


**SAVE $65!**
13" MacBook™
2GHz with 2GB SDRAM, 160GB HD and SuperDrive
FREE Parallels Desktop!*
$1294 - $65= **$1229!*** #7684018 *After mail-in rebate.


**Apple® iWork™ '09**
*Word processing, spreadsheets and presentations that put the fun back into your work!*
only **$74!** #7732910

**LaCie 1TB d2 Quadra External HD**
*eSATA 3Gb/s, FireWire 800, FireWire 400 and USB 2.0!*
only **$174.94!** #7702561

Authorized Reseller

**MacMall**
*Your #1 Apple Superstore!*

**Source code: MACTECH**

Call 1-877-233-2838 or visit macmall.com

*CASH BACK-Purchase select computer models from MacMall and receive up to $250 cash back via MacMall mail-in rebate. Ends 5/17/09. • FREE PARALLELS DESKTOP OFFER-Get Parallels Desktop 4.0 for Mac free after $20 mfr. and $60 MacMall mail-in rebates with purchase of any new Apple computer. Price before rebates is $80. Ends 5/17/09. • ALL OFFERS VALID WHILE SUPPLIES LAST. Download rebate coupons at www.macmall.com/rebates. For rebate terms and conditions, please visit our Web site and enter the applicable part number. Although we do our best to achieve 100% accuracy, occasionally errors and inaccuracies do occur. Should you encounter an error or inaccuracy, please inform us so it can be corrected.